Apache Administrator Training - 3 day

Rich Bowen, Cooper McGregor, Inc

February 24, 2004

Contents

Part I

Day One of Three

Table of Contents

Chapter 1

History

http://www.apache.org/history/

A rather incomplete attempt to provide some kind of Apache timeline and/or historical record. This project was started by Rich Bowen and Thomas Eibner, and is moving along very slowly as time permits.

This section is an attempt to acquaint the student with the history, as well as the historical roots, of the Apache project - how it came to be, why it came to be, and how it has progressed. Additionally, we attempt to give a little bit of context of the surrounding people and projects which shaped the web, and the Apache project.

Finally, we try to give some understanding of the Apache Software Foundation, and what its goals are.

1.1 Pre-history

- Vannevar Bush - As We May Think
- Internet - 1968
- Gopher, FTP, etc

Apache came into existence 4 years after the creation of the World Wide Web. The Internet had been around for a while by then, and frameworks such as Gopher were already in place and in widespread use. But the ideas that formed the Web had been around for at least 45 years.

In 1945, **Vannevar Bush** wrote a paper called **As We May Think**, in which he discusses the way that we think, the way that our minds move from one topic to another, and the ways that technology needed to evolve to service the way that we think. Reading his thoughts on this matter, couched in terms of the technology that was then available, is provides interesting insights into our own time. You can obtain this entire document at http://www.theatlantic.com/unbound/flashbks/com

When the Internet came into existence (1968 is probably a useful date to remember for this), it was more about communication (ie, interpersonal) than about being an information archive. It wasn't until 1990 that these ideas came once again to the forefront, and information dissemination became a major thrust of Intenet usage. And so the WWW was born.

- 1990 - WWW - Tim Berners Lee

  - Working at CERN (European Center for Nuclear Research)
  - Wanted to provide a more efficient way for scientists to access information/documents and "link" from one idea to another
  - Developed `www` client software, the CERN web server, and coined the term "World Wide Web"

- 1992-1994 - NCSA (National Center for Supercomputing Activities) HTTPd

  In 1993 Marc Andreessen released the Mosaic graphical web browser, which you can read about at `http://www.webhistory.org/`. He found that the CERN code was icky, and he asked Rob McCool to write a new web server which was simpler and easier to work with than CERN (according to Rob). I spoke with Rob about this via email, but missed the chance to speak with him at ApacheCon because he was only there for a day. Anyways, at this point, NCSA HTTPd was born, and was maintained by NCSA for a few years Then, in 1995, Netscape Communications was formed, taking most of the programming talent from the NCSA HTTPd project

1.2 Apache

- 1995 - Apache project begins
  - Rob McCool is part of the exodus to Netscape, leaving HTTPd unmaintained
  - Many people using NCSA on their web sites, and many of them had been submitting patches and bug fixes to Rob. Now there's nobody to send them to.
  - Brian Behlendorf, and a group of 7 other developers, coordinates collection of patches (`http://httpd.apache.org/ABOUT\`
  - Name "Apache" proposed, among numerous others. (`http://www.geocrawler.com/mail/thread.php3?subject=name&`
  - Randy Terbush puts together Apache Software License, based on BSD Software License, ensuring that the software will be free and open
  - C2.net donates server space for Apache.org
  - April 1995 - Apache 0.6.2 released
  - December 1995 - Apache 1.0, a complete rewrite, released.
    * Main advance here is modularization of code.
    * Project codenamed "Shambala"
    * Robert Thau main developer of this code
    * Same basic code base in use today in 1.3.x
  - 1997 - Open/Free software goes mainstream
    * 1997 - The Cathedral and The Bazaar
    * Apache deal with IBM - Apache forms codebase for WebSphere
    * Apache Software Foundation formed
    * Revision of License to be more palatable to IBM - Advertising clause removed
    * 1997 - Apache 1.3 released, with Windows support
  - May 2000 - Apachecon Orlando - Apache 2.0 alpha released
  - October 2000 - ApacheCon Europe - Douglas Adams speaks, one of his final speaking ops before he suddenly died.
  - April 2001 - ApacheCon Santa Clara - Apache 2.0 initial beta release
  - April 2002 - Apache 2.0.35 releases as GA (General Availability). 2.0.36 follows shortly after with some important fixes.
  - May 2002 - Apache 1.3 enters maintenance mode (No new features, just bug fixes and documentation updates).
  - `http://uptime.netcraft.com/up/today/top.avg.html`

1.3 Apache Software Foundation

The ASF was formed for a number of reasons. The catalyst was the IBM deal, and their desire to deal with an actual legal entity. However, the impact of creating the ASF was rather larger than that.

The goals of the ASF are:

- provide a foundation for open, collaborative software development projects by supplying hardware, communication, and business infrastructure;
- create an independent legal entity to which companies and individuals can donate resources and be assured that those resources will be used for the public benefit;
- provide a means for individual volunteers to be sheltered from legal suits directed at the Foundation's projects; and,
- protect the 'Apache' brand, as applied to its software products, from being abused by other organizations.

The goal of this section is to help people understand the structure of the ASF. Understanding that Apache is developed by a group of volunteers is a useful thing in that it helps you understand why certain things have priority and others do not. Volunteers work on the things that interest them, and they work on them for as long as it interests them. There is no particular release schedule, and there is no particular external incentive to get anything done if it's not fun.

The ASF exists to give some kind of oversight, legal protection, and quality assurance. Legal protection is probably the most important of these, but we do want to assure that projects under the Apache name are of a certain quality, and that projects that are under the Apache umbrella don't do things that embarass us.

Chapter 2

Installing

http://httpd.apache.org/docs/install.html

In this section, the students will each have to install Apache themselves. They should each have a server system which does not have Apache installed. THey should download the Apache source, verify the pgp signature and MD5 sum, unpack and install from source. They should experiment with the arguments to ./configure and should install Apache with and without DSO support. If the machines are reasonably fast, they should try multiple installations, and see what happens.

Students should install Apache 1.3 and Apache 2.0, and should verify that both are functioning correctly, at the same time, running on different ports.

Students should then reinstall Apache 1.3 using Apache ToolBox, enabling any modules which you wish to use in the remainder of the course. In particular, you will probably want to enable mod_ssl, mod_perl, and mod_php.

Make sure that Apache 2 is installed with DAV and dav-fs enabled.

2.1 Building from source

Although many Unixes come with some version of Apache preinstalled, there are many arguments for installing the server yourself.

- Exactly the way you need it
- Optimized for your hardware
- Ensure nothing strange added
- Directory structure that makes sense to you

This is a good point to encourage discussion of package management systems, particularly if your students are already fond of a particular unix distribution that is tied to a package management system. The arguments for installing from source seem rather weak when weighed against the convenience of a package management system. However, there are times when it is necessary to install from source, and so it is useful to know how to do it.

Get source from http://httpd.apache.org

Latest releases are 1.3.27 and 2.0.45.

You will want to update these numbers with each release of Apache so that you are actually listing the latest version. You will want to make the following two notes about the Apache software distribution site.

First, that the site is mirrored around the world, and that downloading Apache from a mirror site is of great benefit to the ASF, financially. Our monthly bandwidth bill is astronomical.

Next, that you really need to verify the distribution using the MD5 sum, and, if possible, the PGP signature that are available from the site. However, you should get these from the official Apache site, rather than the mirror site. The rationale here is that if the distro was compromised, the signature files probably were also.

Verify the distribution

There are two ways to verify the distribution. The MD5 sum can be used as follows:

```
md5sum apache_1.3.27.tar.gz
```

Verify that the output of that command matches the contents of the .md5 file.

To verify the pgp signature, you will need to first import the keyring from the Apache site:

Download `http`//www.apache.org/dist/httpd/KEYS":

```
wget http://www.apache.org/dist/httpd/KEYS
gpg --import KEYS
```

Finally, verify the signature with:

```
gpg --verify apache_1.3.27.tar.gz.asc
```

Unpack the distribution

`tar -vzxf httpd-1.3.24.tar.gz`

Note that the -z flag is a gnu-tar thing, and may not be available in all versions of tar, although it is more common now than it used to be.

Change into the directory

Build it ...

```
./configure --prefix=/usr/local/apache
make
make install
```

Note that if you run `./configure` as a non-root user, Apache will be configured to run on port 8080. This is very annoying, but being aware of it mostly solves the problem. Tell students that they really only need to be root in order to `make install`, and then see what happens. This is mostly for your own information, since at least one student will do this, and then you'll be left wondering why it's not working.

2.2 Contents of distribution file, 1.3

We now take a brief step back to look at the contents of the distribution file, so that the student knows what we're working with. You'll need to familiarize yourself with the contents of each of the directories in the distribution, so that you can explain any files that they may ask about. Note that the layout for 2.0 is different from that for 1.3, having been reorganized for a variety of reasons.

  - cgi-bin
    * Sample CGI programs
    * Usually not executable, for security reasons
      The rationale here is that, if every web server on the planet has a particular CGI program installed and enabled, and some day, for some reason, someone finds a security exploit in it, it will be a simple matter to crack any server on the planet. Thus, we ship sample cgi programs, but don't enable them by default, and recommend that they be used for testing only.
    * `--cgidir=DIR`

- conf
  - Starter configuration files
  - Will not overwrite your existing configs That is, if you modify your configuration file, and then reinstall Apache, your changes will be preserved. This is particularly important for this exercise, as, after installing Apache the first time, we want to install it again, perhaps several times, with a very different configuration. Thus, each time, we want to remove our configuration file before installing Apache again. Make sure that you explicitly talk about this at this point, as at least one student will forget to do this, and you need to have mentioned it so that you don't look like this problem caught you off-guard.
  - Variables filled in based on your build arguments

    ```
    ServerRoot @@ServerRoot@@
    LockFile @rel_logfiledir@/accept.lock
    ```

    Show the students these variables in the pre-install configuration file, and demonstrate to them how these variables are filled in after the fact. Demonstrate this with several different arguments for the various directories, or with a –with-layout= flag.
  - `--sysconfdir=DIR`

- htdocs
  - "It's working" page
    This page is available in 20 or 30 languages. You should get the correct one, based on your browser preferences. You might want to demonstrate this as a preview of the content negotiation chapter.
  - Manual
  - `--manualdir=DIR` Specifying the –manualdir flag will cause the entire user manual to be placed elsewhere, and an Alias to be inserted in the configuration file pointing to that location

- icons
  - Icons used in auto directory listings
  - `--iconsdir=DIR`

  The manual and the icons are the only directories of content that are generally installed by default. icons is always installed outside of the document directory, but the manual is usually installed inside the document directory. There was an attempt to change this, several versions ago, and move the manual outside of the document directory by default. For some reason, this confused people, and it was made an option, with the default being inside the document directory. Thus, the `--manualdir` flag is fairly new.

- logs
  - Initially empty

- src

- Source code for Apache server
- Subdirs for a variety of things - main, modules, helper apps, etc

It would be useful at this point to demonstrate modifying a source code file, and rebuilding Apache with the change. The recommended example follows:

Apache 1.3

`src/include/httpd.h`, line 429 (in version 1.3.27, anyways)

Modify the name of the product (SERVER_BASEPRODUCT) to be something else, like "Harry's Happy HTTPd", then rebuild. Demonstrate what a HEAD request returns now.

Note that this is just a gimmic, not a security measure. See `http://httpd.apache.org/docs/misc/FAQ.html#serverheader` for more discussion on this matter.

2.3 Contents of distribution file, 2.0

Mostly the same, somewhat different layout.

Root-level directories:

- build
- docs
- include
- modules
- os
- server
- srclib
- support
- test

`docs` contains several subdirs that used to be elsewhere:

- cgi-examples
- conf
- docroot
- error (Error documents)
- icons
- man
- manual

These were placed in a docs/ folder primarily so that the documentation team could have one cvs checkout on which to work. In other words, it got moved because I complained. It's nice to have a little influence sometimes! ;-)

The cgi examples and the default configuration are, technically, considered to be the realm of the documentation team, because they are about best practices.

2.4 Running configure

The `configure` script, located in the root directory of the Apache distribution, configures your Apache compilation.

```
./configure --help

./configure --prefix=/home/httpd --show-layout

./configure --prefix=/usr/local/apache --enable-module=most
  --enable-shared=max

./configure --prefix=/usr/local/apache --enable-module=speling

./configure --prefix=/usr/local/apache --enable-module=speling
--enable-shared=speling
```

You need to run each of the above commands on the big screen, and explain to the students what each of these commands do. Of particular importance are the lines relating to shared objects, as this will come up repeatedly later. When you are done with this section, you really should end up with an Apache installation where everything is a so. We will later build a module with apxs, and swap out the so, to show how easy this is.

You should also demonstrate the config.layout file, and building Apache with a different layout. This should be demonstrated with the –show-layout flag, rather than doing an actual install. You will find that installing everything in /usr/local/apache, as the default setting, will give you less to clean up afterwards, and make it easy for everyone to find things that they need to find during the class. It is important for everyone to have files in the same place, in order for your examples not to confuse the folks who put it somewhere else.

2.4.1 Building with mod_perl

```
gunzip mod_perl-1.xx.tar.gz
tar -vxf mod_perl-1.xx.tar
cd mod_perl-1.xx
perl Makefile.PL APACHE_SRC=../apache_1.3.20/src
  DO_HTTPD=1 USE_APACI=1 EVERYTHING=1
make
make install
```

2.4.2 Other, more complex installs

README.configure is a wonderful resource for installing strange and wonderful combinations of modules. And any 3rd party modules should have detailed instructions for building them.

You should read `README.configure` at least once through, to see the sorts of things that are mentioned in there. Going through one of the example installs in there, particularly one for module(s) that you're not familiar with, can be a very educational experience.

2.4.3 Apache ToolBox

http://www.ApacheToolBox.com/

- Automates build process for any combination of modules.
- Downloads libraries, modules, other stuff, that you don't already have installed
- Has to be updated for each new rev of Apache, so can be behind a little

Apache Tool Box is a shell script which automates the process of downloading and building packages. It is useful in that it prevents you from making silly typing errors, and in that it knows how to install modules with strange requirements. Also, it will download any prerequisites that you don't have, and verify the signatures on those files. This makes (usually) for a very easy installation.

Note!

When we reinstall from Apache ToolBox, make sure that you remove/move/backup your configuration file, as `make install` will not overwrite your configuration file.

2.4.4 2.0

```
./configure --help
```

The 2.0 configure processes uses GNU autoconf, rather than the home-rolled thing that 1.3 used. So the `configure` process will look much more familiar to people used to building Unix software.

For our purposes, we will probably:

```
./configure --enable-module=most --enable-shared=max
```

Or perhaps

```
./configure --enable-dav --enable-dav-fs
```

Since that is about the only 2.0 module that we really have time to get to in this course.

- Change port number to 90
- Change `User` and `Group` directives to be valid
- Start it up

Students will of course need to configure the second web server to run on an alternate port. Let them try to start it up without making these changes, so that they can see the error messages that are generated, and learn to know what they mean when they see them again.

Chapter 3

Starting and Stopping

http://httpd.apache.org/docs/invoking.html

http://httpd.apache.org/docs/stopping.html

3.1 Tangent - Apache architecture

It is useful to understand the Apache architecture before we go much farther.

While this may seem like a bit of a tangent, this is by far the best place to put this in, and helps understand some basic things about how Apache uses your system. It also helps understand why you have to start it as root, but why this is still secure, as well as numerous other things which will be useful in the long run.

Students will frequently contact you after the class is over and tell you how this section helped them fix other problems, not only with Apache, but with IIS, because they had a better understanding of how things were working behind the scenes.

This section is also crucial for understanding many future sections, such as performance, the `User` and `Group` directives, and security.

3.1.1 Apache 1.3

Apache 1.3 is a prefork model - one parent process which manages multiple child processes. More children are forked as needed, and killed off when not needed.

On Win32, a single parent, single threaded child model is used, due to no concept of forking.

There are pictures of this in the next few pages, but you may wish to draw some pictures on the board, as well as encouraging students to look at their process lists, to see what processes are running. Note that by this time they should be running both Apache 1.3 and 2.0, so they will see a strange assortment of processes running. Use the -f flag to gnu ps to show how these processes relate to each other. ps xf, or something like that, depending on what version of ps you are running.

3.1.2 Apache 2.0

Apache 2.0 introduces the MPM model

The MPM - Multi Processing Module - is a way for the particular multi-processing technique of a given platform to be abstracted out. Two examples of multi-processing are threads and forking, and these are two of the available MPMs. With Apache 1.3, this code was contained in if blocks, which were long, icky, and confusing. With 2.0, they are moved out into modules, and you pick the one that is most appropriate for your particular needs and platform. On Unix, you have a number of choices. On non-Unix systems, you are usually limited to a single choice.

- Prefork
- Worker
- Perchild
- Win32
- OS/2
- Netware
- Various others

At this point, you may wish to rebuild Apache 2.0 using the worker MPM, so that you can see the difference that it makes in your process list, if nothing else. Since it takes a significant time to rebuild, you may want to do this and then send folks on a coffee break while it compiles.

```
./configure --with-mpm=worker

./configure --with-mpm=prefork
```

prefork

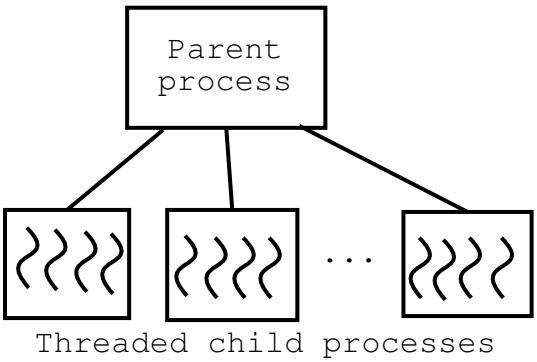
http://httpd.apache.org/docs-2.0/mod/prefork.html

- This is the default
- Looks just like Apache 1.3
- Very robust, but perhaps slower than worker

Robust, because a crash takes out one connection only. However, creating new child processes is slower and more expensive than creating new threads. And, since they take up a bigger memory footprint, you can run fewer of them - hence, less scalable.

worker

http://httpd.apache.org/docs-2.0/mod/worker.html



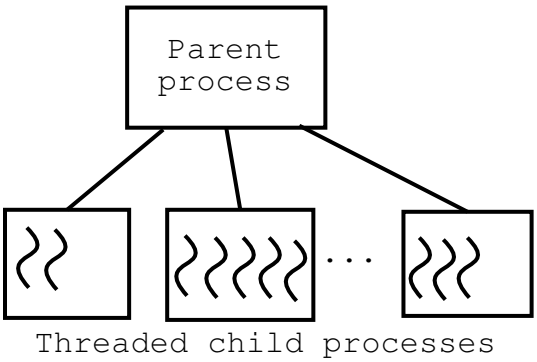
- Multi-process, multi-threaded
- Each child, fixed number of threads
- Launch, reap child processes to deal with changes in load
- ThreadsPerChild

Faster, and less memory use. But less robust because a crash takes out a large number of connections, and so a single problem affects many users. This is the most-recommended MPM, with the following caveats:

- Threading does not work very well on some platforms, like FreeBSD
- Some modules, like mod_php, don't work very well in a threaded environment, and so you need to stick with prefork if you're using php.
- Of course, some modules just don't work at all on 2.0 yet, and in that case you have to stick with 1.3

perchild

http://httpd.apache.org/docs-2.0/mod/perchild.html



- Does not work yet
- Multi-process, multi-threaded
- Allows configuring things per child process

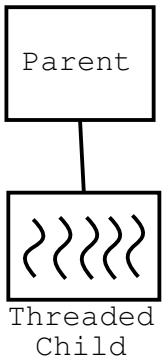
Well, whatever you've heard, don't expect to have a functioning perlchild MPM any time soon. Without going into the whole story, the basics are that the guy that was most actively working on perchild is no longer working on Apache at all, and there has been no progress on the perchild project for several months.

But, in a nutshell, here's what perchild does, and why it is cool:

perchild lets you configure Apache per child. Hence the name. In other words, you can actually have a different configuration for each Apache child process.

This allows you to run virtual hosts as a particular user (as opposed to just the cgi programs, like suexec lets you do). It lets you configure one vhost to run 10 threads, and another to run 200 threads. To specifically assign a particular child process to a particular vhost. And so on. It is very cool. But we should probably quit talking about it, since it is largely imaginary, and likely to remain so for some time.

win32



- Windows only
- Way faster than Apache 1.3 on Win32
- Uses completion ports for additional performance

Clearly, this course could use more meat on the Win32 sections. It would be nice to have at least screen shots of a Win32 installation, and the Apache Monitor thingy that they've added. I'm ashamed to admit that I now know almost nothing about Apache in Windows, having given a number of presentations at ApacheCon about it in the past.

3.2 apachectl

- start
- stop

  - Signals parent process
  - Parent kills child processes
  - Kills self
  - If you just kill child processes, parent will respawn them

- restart
  - Unceremoniously kills child processes
  - Re-reads config file
  - Respawns child processes
- graceful

  Like `restart`, but waits for each child to finish what it is doing.
- startssl
  - Starts SSL
  - Uses -D argument
  - Talk about this when we get to SSL
- configtest
  - Checks conf files
  - Talk about this when we get to configuration
- status
  - Checks status
  - Need to have mod_status installed
  - Talk about this when we get to handlers

  Demonstrate apachectl status, which you should already have configured. However, it will probably not work for the students on their default install systems.
- help
- startssl (If you have mod_ssl installed)

apachectl symlink

You may want to:

```
cd /usr/local/bin
ln -sf /usr/local/apache/bin/apachectl ./
```

so that apachectl is in your path. This will make life easier throughout the rest of the course. Make the students do this. It's still early in the course, and they won't do things if you just suggest it. This will make your life easier later in the course, for those students that don't have a lot of Unix experience. Trust me on this one. Besides, this is about the point in the course where you need to start forcing the students to follow along and do all of the examples, or you will wear yourself out, and get through the material too fast. Note that the MOST common student complaint is that there was not enough hands-on stuff, even when they were the ones that resisted doing the exercises.

3.3 httpd

3.3.1 start

```
/usr/local/apache/bin/httpd

/usr/local/apache/bin/httpd -f /path/to/other/config.file

/usr/local/apache/bin/httpd -DSSL -DOtherVar
```

3.3.2 stop

```
cat /usr/local/apache/logs/httpd.pid | xargs kill
killall -9 httpd
rm -f /usr/local/apache/logs/httpd.pid
```

- httpd.pid contains the PID (process ID) of the parent Apache process

- Each child runs as a separate process

- If you kill children first, parent relaunches them

3.3.3 Other options

```
# /usr/local/apache/bin/httpd -h
Usage: /usr/local/apache/bin/httpd [-D name] [-d directory] [-f file]
                                   [-C "directive"] [-c "directive"]
                                   [-v] [-V] [-h] [-l] [-L] [-S] [-t] [-T]
Options:
 -D name          : define a name for use in <IfDefine name> directives
 -d directory     : specify an alternate initial ServerRoot
 -f file          : specify an alternate ServerConfigFile
 -C "directive"   : process directive before reading config files
 -c "directive"   : process directive after  reading config files
 -v               : show version number
 -V               : show compile settings
 -h               : list available command line options (this page)
 -l               : list compiled-in modules
 -L               : list available configuration directives
 -t -D DUMP_VHOSTS: show parsed settings (currently only vhost settings)
 -t               : run syntax check for config files (with docroot check)
 -T               : run syntax check for config files (without docroot check)
```

Have the students run this command, and experiment with various of the command line options. Encouraging the students to experiment tends to be very hard work, but it is richly rewarding for them later.

3.4 Starting at boot

Various systems will have different ways of doing this. Most Unixes have scripts in `/etc/rc.d` which run on system startup. Somewhere in here, you will want to put ...

```
/usr/local/apache/bin/apachectl start
```

.. in one of those startup scripts, like, perhaps, `rc.local` or `rc.httpd`

Note that some Unixes, like Solaris and RedHat Linux, for example, have a more developed concept of run levels, and you have to work within that framework if you want different things to happen at different run levels.

Also, many Linux'es have rather advanced ideas of what a startup script should or should not do. This can cause confusion in later tech support questions, but knowing how the underlying system works tends to simplify these questions.

Chapter 4

Configuration files

http://httpd.apache.org/docs/mod/directives.html

4.1 The files

Configuration may be in several places:

- `httpd.conf`
  - Used to be several different files
  - `srm.conf`
  - `access.conf`
  - Later merged into one file
  - Now split back out into multiple

  Here's the story. For NCSA, there were three configuration files, and you had to put the right stuff in the right file, or it would not work. These files were called `httpd.conf`, `srm.conf`, and `access.conf`.

  In the early days of Apache 1.3, this restriction got relaxed, and you could put any configuration directive anywhere. While this is nice in some respects, it really is additionally confusing to beginners, who want to know where directives are supposed to go. When the answer is "anywhere you want" they just get confused.

  With 1.3.7, the three config files were rolled into one, and `access.conf` and `srm.conf` became just comments that you should not use them any more. In 2.0, they go away entirely.

  `access.conf` was initially for access-related directives, such as password protection or per-host restrictions.

  `srm.conf` was for Server Resource Management, and so contained stuff about directories, Aliases, and so forth.
- `.htaccess` files
- Include another file
- Include a directory
  - Path relative to ServerRoot
  - All files - watch out for temp files

4.2 Config file syntax

4.2.1 Directives

```
ServerName www.apacheadmin.com
ServerAlias www apache.rcbowen.com apache
```

4.2.2 Anatomy of directive docs

Syntax

A description of the syntax of the directive

Default

The default value, if any. Note that many directives have a setting in the default configuration file, and that this is not necessarily the same thing as the default value.

Context

Where are you allowed to use this directive?

Override

What AllowOverride setting is necessary in order to permit the use of this directive in `.htaccess` files?

Status

What is the status of the module which provides this directive? (Core, Base, Extension, Experimental)

Module

What module provides this directive

Compatibility

Are there differences between different Apache versions that you need to know about?

```
AcceptPathInfo Directive

Description:   Resources accept trailing pathname information
Syntax:        AcceptPathInfo On|Off|Default
Default:       AcceptPathInfo Default
Context:       server config, virtual host, directory, .htaccess
Status:        Core
Module:        core
Compatibility: Available in Apache 2.0.30 and later
```

Please show the class example documentation on the Apache web site, or on your mirror of it, and show them each of these sections. Particularly the Override and Context sections, which are probably the ones that people will most frequently ask about.

4.2.3 Sections

```
<Directory /usr/local/apache/htdocs>
... directives ...
</Directory>
```

May also see them referred to as containers, or scope.

4.2.4 Comments

```
# This is a comment. I like comments.
```

4.3 Sections

- `<Directory>`

  ```
  <Directory /usr/local/apache/htdocs/example>
      Options +Indexes
  </Directory>
  ```

  Restricts the contained directives to the specified directory.

- `<DirectoryMatch>`

```
<DirectoryMatch [Dd]ownload>
    Options +Indexes
</DirectoryMatch>
```

Restricts the contained directives to directories that match the specified pattern. In the given example, any directory that contains the string `download` or `Download` will have the directive applied to it.

- `<Files>`
- `<FilesMatch>`

  Allowing you to be a little more fine-grained, you can specify a particular file or set of files. Note that the `<Files>` directive takes just one file as an argument, which is perhaps not expected, given the name of the directive. To specify more than one file, you can use FilesMatch:

```
<FilesMatch (one|two|three).html>
    SetType text/plain
</FilesMatch>
```

- `<IfDefine>`

  If a particular -D variable is defined, then use this configuration. For example, you might start the server with:

```
    httpd -DUSESSL
```

  Then you could add a config like:

```
    <IfDefine USESSL>
        SSLEngine On
    </IfDefine>
```

- `<IfModule>`

  Checks to see if a particular module is loaded.

```
    <IfModule mod_perl.c>
        SetHandler perl-script
    </IfModule>
```

  Note that the configuration section is completely ignored if the specified module is not loaded.

- `<Limit>`
- `<LimitExcept>`

  Limits directive scope by method. Very seldom useful.

```
        <Limit GET POST>
            order deny,allow
            deny from all
            allow from 192.168
        </Limit>
```

You'll see many Authentication tutorials that tell you to use this syntax when setting up password protection. Ignore them.

- `<Location>`
- `<LocationMatch>`

  Map a URL to a handler. We'll look at Location more when we talk about handlers.

- `<VirtualHost>`

  Much more about this at a later date.

4.4 Options

- `ExecCGI` (See CGI)
- `FollowSymLinks` (See security, performance)
- `SymLinksIfOwnerMatch` (See security, performance)
- `Includes` (See SSI)
- `IncludesNOEXEC` (See SSI)
- `Indexes` (See autoindexing)
- `MultiViews` (See Content Negotiation)
- `All`
- `None`

4.5 Different config file

```
httpd -f /usr/local/apache/conf/other.conf
```

- Multiple Apache daemons
- Test configurations
- Restarting to a backup config when something new breaks

Chapter 5

.htaccess files

5.1 Configuration

5.1.1 AccessFileName

Configures the name of the file that will be looked for in each directory.

5.1.2 AllowOverride

- AuthConfig
- FileInfo
- Indexes
- Limit
- Options
- All
- None

http://httpd.apache.org/docs-2.0/mod/core.html#allowoverride

5.2 Performance

Checks for .htaccess in EVERY directory up to the location of the file that is being served. Possibly a big performance hit.

For example, if you are serving a file out of the directory `/usr/local/apache/htdocs/services/training/apache/tutorial` then Apache will (possibly) look for the files:

```
/.htaccess
/usr/.htaccess
/usr/local/.htaccess
/usr/local/apache/.htaccess
/usr/local/apache/htdocs/.htaccess
/usr/local/apache/htdocs/services/.htaccess
/usr/local/apache/htdocs/services/training/.htaccess
/usr/local/apache/htdocs/services/training/apache/.htaccess
```

Note that this is the absurd worst case, and never happens in practice. In order to make this happen, you'd have to have `AllowOveride All` set for `/` which nobody in their right mind would ever do. On the other hand, we should not assume anything. People unaware of the impact of this might indeed to that for convenience. But in practice, you end up checking for .htaccess files down to whatever lowest level you have permitted `AllowOverride`.

Thus, if you:

```
<Directory /usr/local/apache/htdocs>
    AllowOverride All
</Directory>
```

Then you'd really only get requests for:

```
/usr/local/apache/htdocs/.htaccess
/usr/local/apache/htdocs/services/.htaccess
/usr/local/apache/htdocs/services/training/.htaccess
/usr/local/apache/htdocs/services/training/apache/.htaccess
```

Which is still a little much, but less serious. Note, however, that dropping the `AllowOverride` directive in this scenario gives 2- or 3-fold performance improvement, even for directories where you're not using `.htaccess` files at all, based on my benchmarks. I'd recommend that you do your own benchmarks to verify these numbers.

- Directives found will be applied in the order that they are found, overriding previous settings.
- This is done every time a file is served
- Can have multiple settings for `AccessFileName`

  ```
  AccessFileName .htaccess .acl directory.conf
  ```

  In which case, it will look for each of these files in each directory

5.3 Exercise

Create `.htaccess` file in your document directory, containing the following, or something like it:

```
DirectoryIndex default.htm
```

Create a `default.htm` file in that directory, and get the `.htaccess` file working.

Students seem to require a lot of hand-holding here. They will need to do a number of things to get this working, and the idea is to get them to do as much of it by themselves as possible. They will need to:

Create `default.htm`

Create `.htaccess`

Enable `AllowOverride` appropriately for the directory in question.

Encourage them to use the correct `AllowOverride` setting, rather than just using `All`. Impress upon them the dangers of using `AllowOverride Options`, which is, of course, included in `All`.

Chapter 6

Virtual Hosts

http://httpd.apache.org/docs/vhosts/index.html

A virtual host (vhost for short) is a means of running more than one web site on the same Apache server. This can be done one of two ways:

- IP-based virtual hosts
- Name-based virtual hosts

Fortunately, the procedure is almost identical. There's really just one small difference.

What you'll need today

- Ability to edit hosts file
- Basic understanding of DNS and/or name resolution in general
- Some patience

6.1 IP-based virtual hosts

http://httpd.apache.org/docs/vhosts/ip-based.html

- Requires a unique IP address for each host
- The preferred (only, really) way to do SSL hosts
- A little less confusion as to which vhost gets picked

```
<VirtualHost 192.168.1.7>
    ServerName www.foo.com
    DocumentRoot /home/foo/htdocs

    CustomLog /home/foo/logs/access_log common
</VirtualHost>
```

Note that the docs have a server name, rather than IP address, as the argument. This is a bad idea, in case dns has not come up yet.

Most directives are valid in a VirtualHost section. See the docs for a particular directive to see what context they are valid in.

If you like, you can have students bind a secondary IP address to their network card, and set up virtual hosting that way. Alternately, and much easier, you can have them set up vhosts on 127.0.0.2, 127.0.0.3, etc. This can be entertaining, and often students learn something they didn't know about networking in the process.

Also, setting up IP-based vhosts first tends to help people understand name-based vhosts a little better.

6.2 Name-based virtual hosts

http://httpd.apache.org/docs/vhosts/name-based.html

- One IP address, multiple hosts
- Can't do SSL this way
- Can end up having overlaps and/or conflicts if you misunderstand the configuration

```
NameVirtualHost 192.168.1.7

<VirtualHost 192.168.1.7>
    ServerName www.foo.com
    ...
</VirtualHost>

<VirtualHost 192.168.1.7>
    ServerName www.boxofclue.com
    ...
</VirtualHost>
```

Notes:

- Argument to NameVirtualHost and to VirtualHost must the exactly the same literal string, or it will not know that you are dealing with the same address Best to use `*` rather than a particular address

```
NameVirtualHost *

<VirtualHost *>
    ServerName www.coopermcgregor.com
    DocumentRoot /usr/local/apache/vhosts/cmi
</VirtualHost>

<VirtualHost *>
    ServerName www.boxofclue.com
    ServerAlias boxofclue.com clueful.com
    DocumentRoot /usr/local/apache/vhosts/clue
</VirtualHost>
```

- Can specify a port:

```
NameVirtualHost *:443
NameVirtualHost 192.168.1.5:8080
```

- Note that the only real difference between name-based and IP-based (in the configuration, that is) is the NameVirtualHost directive.
- You only need to put directives in one of these sections where they differ from the global setting
- See later section on logging for info about per-vhost logs

Make sure that the class understands that these vhosts are running on the same IP address, and the same port. The only way that the server knows which vhost you want is the `Host:` header that gets sent with the request. This is why old browsers (ie, pre 1996 - nothing to worry about) and telnet requests tend to get the wrong vhost.

6.3 General caveats, comments

- If you set one host up as a vhost, you should probably set all hosts up that way. 50+ % of the vhost problems that we see on #apache are because default configurations are overriding vhost configurations
- Use Listen not Port
- Consider putting each vhost config in its own file. This will save you hassle and confusion later

6.4 Exercise

1. Add 2 or more additional names to your "/etc/hosts" file (virtual1 and virtual2)
2. Set up a virtual host for each one (put vhosts in /usr/local/apache/vhosts/`servername`)

3. Verify that they are serving content out of different directories

4. If you run into any problems, ask the folks on #apache for help. They have promised to be nice.

6.5 Additional notes, examples, etc

6.5.1 /etc/hosts

`/etc/hosts` is a file containing mappings from name to IP address for your machine. Entries in the file look like:

```
192.168.1.104 virtual1
```

Once that entry has been added, the name `virtual1` will immediately start resolving to the IP address 192.168.1.104. Note that this is just for you. other people will not see this mapping. You may need to restart your browser to have this take effect, as most browsers cache name records.

6.5.2 Your example virtual host sections

For the exercise above, you should have ended up with a configuration that looked something like:

```
NameVirtualHost *

<VirtualHost *>
    ServerName virtual1
    DocumentRoot /usr/local/apache/vhosts/virtual1
</VirtualHost>

<VirtualHost *>
    ServerName virtual2
    DocumentRoot /usr/local/apache/vhosts/virtual2
</VirtualHost>
```

6.5.3 #apache

#apache, referred to in the above section, is an IRC channel. If you are not familiar with IRC, this may seem a little odd. IRC - Internet Relay Chat - is real-time chat over the Internet. #apache is the name of a "channel" on which people talk about (most of the time) Apache and related topics. #apache is on the `irc.openprojects.net` network.

If you have a IRC client installed (XChat is nice) you can connect to this server, and join this channel. There is usually at least one person there who knows what they are talking about.

6.5.4 mod_vhost_alias

At this point in the course, if we have time, we'll experiment with mod_vhost_alias, which is a module allowing bulk virtual hosting. You'll find a sample configuration in the `examples` directory on your CD, which should look something like this:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%1.1/%1.2/%1.3+/htdocs
VirtualScriptAlias /usr/local/apache/vhosts/%1.1/%1.2/%1.3+/cgi-bin
```

As this is primarily a hands-on experiment, little space is given to this in the notes. Please see http://httpd.apache.org/docs-2.0/mod/mod_vhost_alias.html for more information.

Students tend to want more experimentation here, so I've attempted to add some better examples that they can play with. Have them try one or more of the following:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%2/%1/htdocs
```

Which should map `www.foo.com` to `/usr/local/apache/vhosts/foo/www/htdocs` and `bob.foo.com` to `/usr/local/apache/vhost`

or ...

```
VirtualDocumentRoot /usr/local/apache/vhosts/%1.1/%1.2+/htdocs
```

Which should map `www.foo.com` to `/usr/local/apache/vhosts/w/ww/htdocs` and `bob.foo.com` to `/usr/local/apache/vhosts/b`

Note that in order to actually try these things, they will need to create a lot of directories and add a lot of hostfile entries. This is only useful in practice for wildcard DNS entries.

Chapter 7

MIME

http://httpd.apache.org/docs/mod/mod_mime.html

MIME - Multipart Internet Mail Extensions - was created in order that attachments could be sent via email, rather than having email restricted to plain text only. HTTP is built around MIME, and headers in general. Perhaps this section would better be labeled **HTTP Headers**.

- Multipart Internet Mail Extensions
- RFC 2045 - 2049
- HTTP based entirely on MIME standards
- MIME header tells the browser what type of document it is getting
- Content-Type: major/minor
- Content-Type: text/html
- Content-Type: image/gif
- Content-Type: Application/Unknown
- The browser has a list of mappings to applications, so that it knows how to display the content.

7.1 HTTP headers

Slight rewind here - HTTP is all about headers. Most of the information about a transaction is contained in the headers. The body is actually quite uninteresting (at least from a protocol perspective).

http://webtools.mozilla.org/web-sniffer - Tool for viewing the complete HTTP transaction, including all headers.

Or, try `http://www.web-caching.org/showheaders.html`

or, `http://www.mdb.ku.dk/tarvin/http_tool`

- Content-Length
- Content-Encoding
- Location
- Format is Header: value
- Headers are not case-sensitive
- Headers are terminated by a blank line

```
Header: value
Header: value
Header: value

Body here
```

- The message is over when `Content-Length` bytes have been served.
- On dynamic documents, either the size is calculated before the document is delivered, or it is delivered in chunks, with a `Content-Length` header on each chunk. (This is called ”chunked encoding”.)

7.2 MIME configuration

TypesConfig

- conf/mime.types
- Maps types to file extensions
- audio/x-realaudio ra
- video/mpeg mpeg mpg mpe
- Don’t add mime types here
- Not case sensitive, dot not required

The reason that you don’t edit the TypesConfig file is that on “make install” the TypesConfig file is overwritten, always, no matter what changes you have made to it. Changes may have been made to this file at the standards-board level, and you need to get those new file types, or whatever. So you should make your changes using the directives below to alter the MIME types mappings.

7.2.1 AddType

- AddType image/png .png
- `AllowOverride FileInfo` lets you put these in .htaccess files

```
AddType application/x-tar tgz
```

Note that a surprising number of students will want to get pedantic about the "x-" in the mime type above. It's not clear why it is there, but browsers recognize it, and expect it. Although Internet Explorer will also work without it it, some older versions of Netscape will not.

7.2.2 RemoveType

- Removes a mapping that was previously in place

  The scenario here is a site that has a cgi directory with examples of the software, and then a download subdirectory containing the source code, so that people can download and examine the code. While better directory organization might be more in order, this at least illustrates the concept. See also RemoveEncoding.

```
<Directory /www/docs/products>
Options +Includes +ExecCGI
AddType application/x-httpd-cgi cgi
</Directory>

<Directory /www/docs/products/conference>
Options -ExecCGI
RemoveType cgi
</Directory>
```

- Removes the mapping
- Remember that directives trickle down through the directory tree unless explicitly overridden like this

7.2.3 DefaultType

- This type is used unless another is explicitly set
- Core directive, not a mod_mime directive.
- text/html by default, if not set

7.2.4 ForceType

- Sets the MIME type for all files in the scope, regardless of filename
- ForceType image/gif

Example: Images uploaded from digital camera called `dc00034`, `dc00035`, `dc00036`, etc, without a file extension. Rather than having to rename all the files to `something.jpg`, I can just:

```
<Directory /usr/local/apache/htdocs/photos>
    ForceType image/jpg
</Directory>
```

7.3 mod_mime_magic

- Determines file type based on content of file.
- Runs the `file` program to determine
- magic.conf contains mappings to mime types

7.4 Encoding

- Usually compression
- Can be other encoding, such as uuencode
- Is additional to content type
- Can have multiple file extensions to convey this information
- resume.doc - Microsoft Word document
- resume.doc.zip - PKZipped Microsoft Word document
- The default is that a file is sent as is, with no encoding

7.4.1 AddEncoding

- Adds an encoding mapping to a particular file extension
- AddEncoding pkzip .zip
- AddEncoding gzip .gz

7.4.2 RemoveEncoding

- RemoveEncoding gz
- Removes any encoding that has been associated with the specified file extension

This directive has been useful on the Apache download site itself. Files with a .tar.gz file extension should probably not be sent with a gzip encoding, as this will cause them to be uncompressed upon arrival, which is typically not the desired behavior. Likewise, on the Apache site, we want .tar.gz.asc files that contain the gpg signature for the corresponding .tar.gz file, but are not themselves either a .tar or a .gz file. Thus, when sent with a gzip content encoding, they arrive as a zero-byte file, since there is no valid gzip content in the file.

7.4.3 mod_gzip

More will be said about mod_gzip later. Files are compressed as they are sent out to the client, and an additional `Content-Encoding` header is attached to the file to let the browser know that the content needs to be decoded (uncompressed) before it can be displayed.

7.5 Language

- Content-Language header specifies the language that is being sent
- Browser configuration can determine the preferred language (See Content Negotiation)
- Can be set in addition to other attributes

7.6 Multiple file extensions

Files can have more than one file extension in order to convey more than one of the above pieces of information.

```
file.tar.gz.en
file.tar.Z.fr
file.html.gz.de
```

There's no great value of going into much detail here with the language stuff, as we will get into much greater depth in the Content Negotiation section, which is just a few down the line.

You may also wish to make a note that multiple file extensions conveying the same piece of information will cause all but the last one to be ignored. Thus file.doc.tar.txt is a text file, not a doc or tar file.

7.7 Experiment

- In your vhost directory, create a file called `something.abc`
- Add a MIME type to this file extension
- Verify that your browser loads it with this MIME type, and asks you what it is supposed to do with it.

The purpose of this exercise is to illustrate that you can make up your own file types, and cause the browser to behave a certain way upon receiving that file type. This is how people come up with custom plug ins, file handlers, or whatever. You have some client-side application that gets mapped to a particular MIME type header.

Note also that Internet Explorer occasionally thinks that it is smarter than you. That is, it will, in some conditions, follow the file extension rather than the MIME type. Thus, a txt file with an Application/Unknown MIME type (in order that the user will be forced to choose a file location to download and save the file, for example) may in fact be displayed in the browser as plain text by IE, which follows the file extension rather than the MIME type.

While this is clearly undesirable behavior, you should note that somewhere between 70 and 98% of your audience will be running IE, and you must plan accordingly.

Fortunately, this is not a web design course. However, this is a reminder to test with several browsers in order to assure correct behavior everywhere.

Chapter 8

Logging

Every request to your server results in an entry in a log file. If something goes wrong, it will also result in an entry in the error log file. This means that you always have a "paper trail" for everything that goes on on your server, so that you can look back and find out what happened. This is primarily useful for two purposes - troubleshooting, and statistics gathering.

In this chapter, we look at the standard log files, as well as at the custom log files which you can create to fit non-standard needs.

8.1 Standard log files

`/usr/local/apache/logs/access_log`

First we look at a standard default access log, in Common log format (CLF). While many third-party bundlings of Apache ship with the Combined log format instead, this is still the most common log format, and the Combined is just an extension of it.

8.1.1 access_log

Format ...

```
216.35.116.91 - - [19/Aug/2000:14:47:37 -0400] "GET / HTTP/1.0" 200 654
```

- `216.35.116.91` Client address (See HostNameLookups)
- `-` (placeholder) Ident (always blank)
- `-` (placeholder) Username
- `[19/Aug/2000:14:47:37 -0400]` - Date/time

- `"GET / HTTP/1.0"` Request
  - Method type (GET, POST, HEAD)
  - URL requested
  - PROTOCOL (HTTP + version number)
- `200` Status code
- `654` Bytes transferred

- Client address - This is the IP address of the client connecting to your server. There are a number of comments that should be made about this.
  - It is the IP address, not the host name. If you turn on HostNameLookups, you'll get the host name, rather than the IP address. Don't do this, for reasons that will be discussed in the performance section.
  - Some folks will get rather irate that you have this information. Of course, this is silly, as the information is probably worthless. However, there are a variety of good reasons to keep your log files highly confidential. Releasing this information (this IP address visited this page at this time) can cause embarassment, or worse. The log files are useful as statistical information, but individual entries should not be public information.
  - Note that many hosts may be behind one address, and, conversely, one host can show up, at different times, behind many IP addresses.
- Ident - This field is here for historical reasons. Long long ago (Netscape 0.9) this field would contain the email address of the person visiting your web site. The browser happily provided this information along with every request. As you can imagine, the marketing people got hold of this information, and started sending UCE to those addresses. Browsers quickly stopped providing this information, and this field has been blank ever since.

  There is a patch available at http://mm.apache.or.jp/pipermail/apache00-01/2000-July/001181.html which adds a directive (Anonymous_Email_As_Ident) which causes the "password" field supplied to be logged in the Ident field. This makes a lot of sense, actually, and we'll come back to that when we get to the Auth chapter.
- Username - Set only if the resource in question required authentication, and, in that case, will contain the username of the authenticated user.
- Date/Time - The date and time that the request was made. Granularity is 1 second, and, no, there's no way to make that finer.
- Request - Contains the full request as received from the client. This should contain three fields, such as: "GET / HTTP/1.0". The first field - the method - can be one of a dozen or so methods such as GET, POST, HEAD, CONNECT, PROPFIND, and so on. The next field is the requested URL. For a local URL, this will be the path - that is, no hostname. For proxied URLs, it will be the full URL of the remote resource. Finally we have the protocol and version, such as HTTP/1.0 or HTTP/1.1.
- Status Codes:

100	Informational
200	OK
300	Redirect
400	User error
500	Server error

- Bytes transferred

8.2 Location and format of the log file

```
CustomLog /usr/local/apache/logs/access_log common
```

Means that the log file is to be located at /usr/local/apache/logs/access_log and is to be in the format 'common'. This format is defined by the LogFormat directive:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

See pages 334, 335

Other log formats:

- Common Log Format (CLF) `"%h %l %u %t \"%r\" %>s %b"`
- Common Log Format with Virtual Host

  ```
  "%v %h %l %u %t \"%r\" %>s %b"
  ```

- NCSA extended/combined log format

  ```
  "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
  ```

- Referer log format `"%{Referer}i -> %U"`
- Agent (Browser) log format `"%{User-agent}i"`

The CustomLog directive can then be used to create a new log file using the format that you have created.

You can have as many log files as you like.

8.3 Exercises

1. Construct a log file containing the status of the request, the virtual host from which it was requested, and the request protocol
2. Construct a log file containing the URL requested, and the referer, but which does not record anything if there was no referer.
3. Construct a log file containing the hostname and port number that the request was served from, in addition to the other information contained in the common log format.

8.3.1 Error logs

Location of the error log

```
ErrorLog logs/error_log
```

Note that path is Relative to ServerRoot

8.3.2 LogLevel

- emerg
- alert
- crit
- error
- warn
- notice
- info
- debug

8.4 Typical errors

- Document error

  ```
  [Fri Aug 18 22:36:26 2000] [error] [client 192.168.1.6] File does not
  exist: /usr/local/apache/bugletdocs/Img/south-korea.gif
  ```

  As with access_log, error message is in several distinct parts

- Authentication error

  ```
  [Tue Apr 11 22:13:21 2000] [error] [client 192.168.1.3] user rbowen
  authentication failure for "/cgi-bin/hirecareers/company.cgi":
  password mismatch
  ```

- CGI errors

```
Wed Jun 14 16:16:37 2000] [error] [client 192.168.1.3] Premature
end of script headers:
/usr/local/apache/cgi-bin/TestProg/announcement.cgi
Global symbol "$rv" requires explicit package name at
/usr/local/apache/cgi-bin/TestProg/announcement.cgi line 81.
Global symbol "%details" requires explicit package name at
/usr/local/apache/cgi-bin/TestProg/announcement.cgi line 84.
Global symbol "$Config" requires explicit package name at
/usr/local/apache/cgi-bin/TestProg/announcement.cgi line 133.
Execution of /usr/local/apache/cgi-bin/TestProg/announcement.cgi
aborted due to compilation errors.
```

8.4.1 Things to remember!

- The error log is your friend
- `tail -f /usr/local/apache/logs/error_log`
- Watch the log while you are working on stuff.

8.5 Logfile reporting

8.5.1 What your log file tells you

- Address of remote machine
- Time of visit
- Resource requested
- What's broken

8.5.2 What your log file does not tell you

- Who is visiting
  - Proxies
  - Caches
- Precisely how many visitors
- Name, address, credit card number
- If you want to know something, you have to ask

8.5.3 Log file parsing

- Webalizer (www.mrunix.net/webalizer)
- Analog
- WebTrends
- WWWStat
- Wusage
- Apache::ParseLog Perl module

8.6 Logging to a process

```
CustomLog |/usr/bin/apachelog common
```

Logs to the program /usr/bin/apachelog, rather than to a file, where apachelog looks like ...

```
#!/usr/bin/perl

while (my $log = <STDIN>) {
    DoSomethingUseful($log);
}
```

- Buffering
- Performance
- `http://modules.apache.org/` for modules that already do this

8.7 Logging to syslog

```
CustomLog syslog combined
CustomLog syslog:local1 common
```

8.8 Rotating log files

8.8.1 Logfile::Rotate

Logfile::Rotate is a Perl module. It is on the CD. Or you can get it from
`http://www.cpan.org/modules/by-module/Logfile/`

Unpack it somewhere, run the following:

```
perl Makefile.PL
make
make test
make install
```

Or, to install the module from the CPAN shell:

```
perl -MCPAN -e shell
cpan> install Logfile::Rotate
cpan> quit
```

Then put the following code into a file called rotatelogs.pl

```
#!/usr/bin/perl

use Logfile::Rotate;

$logfile = new Logfile::Rotate(
    File => '/usr/local/apache/logs/access_log',
    Count => 5,
    Gzip => '/bin/gzip',
    Post => sub {
      `/usr/local/apache/bin/apachectl restart`;
    }
);

$logfile->rotate();
```

Run the file a few times and see what happens to your log files. You'll want to run this program every month (or week, or whatever) via cron or other scheduler.

8.8.2 rotatelogs

```
CustomLog "|/usr/local/apache/bin/rotatelogs /some/where 86400" common
```

Implemented as a piped log file, which rotates your logs automatically every day.

The `/some/where` is where you want it to put the old log files each day. Specifically, it will create files called `/some/shere.XXXXXXX`, where `XXXXXXX` is the time in Unix time - that is, the number of seconds since Jan 1, 1970. This number can be converted back into human-readable time by the command:

```
perl -le 'print scalar localtime($time);'
```

where $time is the number appeneded to the end of the file name.

The 86400 is the number of seconds in a day, and means that the log rotation will happen once a day. You may want to set this at 300 seconds (ever 5 minutes) to see what this does.

8.8.3 logresolve

```
cp ../logs/access_log ./
./logresolve -s stats < access_log > resolved.log
```

Resolves IP addresses, generates simple statistics.

8.9 Logging for multiple virtual hosts

- Each vhost should have its own log file
- Alternately, if you have only one log file, make sure you use the extended (combined) log file format which contains the vhost name:

  ```
  "%v %h %l %u %t \"%r\" %>s %b"
  ```

  Note that this is the canonical name for the vhost, not necessarily the name with which the host was accessed. ie, could be www.apacheadmin.com rather than apacheadmin.com even if that's what was actually used.

Part II

Day Two of Three

Table of Contents

Day two covers a variety of configuration-related items.

Chapter 9

URL Mapping

http://httpd.apache.org/docs/sections.html

9.1 URL Mapping procedure

The process of translating a URL into an actual something that is sent out to the user.

While most folks seem to subconsciously assume that URLs map to some file on the server, this is not always the case. In fact, as the web becomes progressively more dynamic, this is less and less the case. URLs map to resources, where the definition of "resource" varies greatly.

The URL mapping phase is when the server tries to figure out what a URL means - ie, what "resource" the URL refers to, whether it is actually a file, or something else entirely.

This section covers the various ways that the server administrator can force a particular URL to map to a particular resource, with the default behavior being to try to look for a file of the specified name.

9.2 Location

- Not tied to file space
- Usually maps to a handler or script

The `Location` container is used to limit the scope of directives to a range of URLs - a subset of "url-space", if you will. It is often used for the purpose of mapping URLs to a particular handler, but this is by no means the only way that it can be used. It can occasionally be used as a substitute for the Directory directive, but the meaning is somewhat different, in that it matches a URL rather than a directory, and so applies to URLs that don't actually map to a directory.

9.3 Alias

- Maps a URL to a directory, often outside of the DocumentRoot directory

Used to map a URL to a directory - usually a directory outside of the main document directory, although this is not necessarily the case. The default distro, for example, comes with an Alias for the documentation (/manual) which points to a directory within the document root. This is so that you can move it if you want to, but is really rather redundant. For one release (1.3.25 perhaps?) /manual actually did move outside of the document root, but it moved back in the next release because it irritated people who don't like change.

The Alias for /icons/, on the other hand, points to a directory that has always been outside of the document root.

Important note about Alias. The slashes must match. That is, if the first argument contains a slash, the second one should also. Thus:

```
Alias /foo /var/www/foo
```

Note that the first argument (`/foo`) has no trailing slash, and, thus, the second argument (`/var/www/foo`) also should not. When the slashes don't match, bad things happen. Alias is taken very literally. The string in the URL is replaced verbatim with the argument provided. This can result in file paths with too many slashes, or two few, depending on which side you erred.

For example, if you have:

```
Alias /icons/ /usr/local/apache/icons
```

You will end up with errors in your logs which say something like:

```
File /usr/local/apache/iconssomething.gif not found
```

Note the missing / between `icons` and `something.gif`. That's your clue that this is what is happening.

9.4 ScriptAlias

- Maps a URL to a directory, and indicates that the directory contains CGI programs

```
ScriptAlias /cgi-bin/ /usr/local/apache/cgi-bin/
```

Equivalent to ...

```
Alias /cgi-bin/ /usr/local/apache/cgi-bin/
<Directory /usr/local/apache/cgi-bin/>
Options +ExecCGI
SetHandler cgi-script
</Directory>
```

9.5 AliasMatch and ScriptAliasMatch

- Just like Alias and ScriptAlias, but with regular expressions

9.6 Regular Expressions

This is a bit of an aside, but is useful for the rest of this stuff to make sense.

Regular expressions are a means of matching arbitrary patterns in text. It can be a very full-featured library of pattern matches. Here's the smaller list of them:

You can actually spend a pretty substantial amount of time on Regular Expressions if you really want to. Here we try to keep it down to the basics, with a little more detail in the next few pages. Apache 1.3 uses the regex engine from egrep (or at least that same one) and Apache 2.0 uses PCRE, which is more full-featured.

.	Matches anything
+	One or more of the previous character
	Zero or more of the previous character
[ ]	Character class - match one thing in here
?	Optional
^	Start of string
$	End of string
^	(Inside a character class) Not

9.7 Redirect

- Maps a URL to an external URL. (Alias is always to an internal document)

```
Redirect /HyperCal.html http://www.coopermcgregor.com/products/hypercal/
```

9.8 RedirectMatch

- With regexes

```
RedirectMatch [sS]upport(.*) http://www.coopermcgregor.com/support/
Redirectmatch [dD]r[Bb]acc?h?us.* http://www.drbacchus.com/
RedirectMatch (.*) https://otherserver.com$1
```

9.9 RedirectTemp and RedirectPermanent

- Generate different redirect codes

9.10 DocumentRoot

- If all else fails, it must be a request for an actual document, so we look in the DocumentRoot for the path requested.

9.11 Error documents

- Maps an error condition to a more useful error message
- ErrorDocument 404 /cgi-bin/404.cgi
- ErrorDocument 404 http://www.errors.com/
- ErrorDocument 500 /errors/500.html
- ErrorDocument 403 "You need to log in first"

Quotes on ErrorDocument

In Apache 1.3, when you specify a string argument to `ErrorDocument`, you start with quotes, but do not close the quotes on the end of the string. In Apache 2.0, you need to close the quotes.

Example of a 404 CGI handler.

In your configuration file, put:

```
ErrorDocument 404 /cgi-bin/404.cgi
```

Then `/cgi-bin/404.cgi` will look like:

```
#!/usr/bin/perl
use Mail::Sendmail;
use strict;

my $message = qq~
Document not found: $ENV{REQUEST_URI}
Link was from: $ENV{HTTP_REFERER}
~;

my %mail = (
To => 'admin@server.com',
From => 'website@server.com',
Subject => 'Broken link',
Message => $message,
);
sendmail(%mail);

print "Content-type: text/html\n\n";
print "Document not found. Admin has been notified";
```

This is a good hands-on exercise for the students, in that it will suggest to them things that they can do in their own environment that will be more useful. The tests can be made conditional, or set up to send batch email rather than one per error, or other things. You may wish to implement a number of these on your own, so that you can display a few example alternatives.

9.12 Error documents in Apache 2.0

Apache 2.0 has a new way of handling ErrorDocument that will mean much more customizable error messages, rather than the same old boring "Document Not Found" errors.

In you Apache 2.0 default configuration file, you will see the following:

```
<IfModule mod_negotiation.c>
<IfModule mod_include.c>
   Alias /error/ "@@ServerRoot@@/error/"

   <Directory "@@ServerRoot@@/error">
       AllowOverride None
       Options IncludesNoExec
       AddOutputFilter Includes html
       AddHandler type-map var
       Order allow,deny
       Allow from all
       LanguagePriority en es de fr
       ForceLanguagePriority Prefer Fallback
   </Directory>

   ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
   ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
   ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
   ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
   ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
   ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
   ErrorDocument 410 /error/HTTP_GONE.html.var
   ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
   ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
   ErrorDocument 413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
   ErrorDocument 414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
   ErrorDocument 415 /error/HTTP_SERVICE_UNAVAILABLE.html.var
   ErrorDocument 500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
   ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
   ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
   ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
   ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var

</IfModule>
</IfModule>
```

In the directory `@@ServerRoot@@/error/` you will find all of those .html.var files, which contain SSI directives for building custom ErrorDocument pages. And, thanks to the efforts of several people, they are avaialable in several languages. These error documents can be customized to your heart's content.

The 404 page, for example, looks like the following, in English:

```
-----------------------------

 <!--#set var="TITLE" value="Object not found!" -->
 <!--#include virtual="include/top.html" -->

    The requested URL was not found on this server.

  <!--#if expr="$HTTP_REFERER" -->

    The link on the
    <a href="<!--#echo encoding="url" var="HTTP_REFERER"-->">referring
    page</a> seems to be wrong or outdated. Please inform the author of
    <a href="<!--#echo encoding="url" var="HTTP_REFERER"-->">that page</a>
    about the error.

  <!--#else -->

    If you entered the URL manually please check your
    spelling and try again.

  <!--#endif -->

 <!--#include virtual="include/bottom.html" -->

-----------------------------------
```

As you have the full array of SSI variables at your disposal, this lets you customize this page as much as you like.

9.13 Other modules that handler URL mapping

9.13.1 mod_speling

- Corrects common typos
- CheckSpeling On

Handles transposition of characters, common mistypes (l instead of 1, o instead of 0, etc) and mis-capitalizations. Makes things run slower, but is very useful in migrating from Windows to Apache, for example.

Also, if there are several possible matches, you will get a listing of choices.

Exercise: Turn on `mod_spelling` checking for your server. Experiment with URL correction

9.13.2 mod_rewrite

- Alter URLs on the fly as they come in

- Will be covered in more detail on day 5 Examples in the slides, but we'll not dwell on them, as they will be discussed in detail later.

9.13.3 mod_userdir and public_html

- URLs that begin with ~ (tilde) map to that user's directory

```
http://www.uky.edu/~rbowen/
```

- UserDir specifies where that home directory is supposed to be

```
# Serve files out of /home/username/public_html
UserDir public_html

# Serve files out of somewhere else
UserDir /www/users/*/htdocs
```

- Be careful with permissions.

  There should be a discussion here about file permissions, home directory security, and why things are the way that they are. Possibly talk about perchild here, although that may be getting a little silly, now that perchild has been untouched for more than 6 months.

  If UserDir is set to public_html, then you must assure that /home and /home/username and /home/username/public_html are all readable and executable by the Apache user (defined in the User directive). The x is necessary in order for Apache to get directory listings.

  If the users are concerned that this creates an unacceptable security situation, then they are paying attention. Good job.

  Users should not put anything directly in /home/userdir, but should put it in subdirectories thereof, and this really should take care of any security concerns that they may have. I've never quite understood why people got so uptight about this.

  On the other hand, the contents of the public_html directory is world readable, which means that even content that is password protected on the web will be fully available to anyone that has an account on the server itself. This can be a concern on systems where multiple users are rending web space. There's not really any way around this, unfortunately.

- Disable for certain users

```
UserDir enabled
UserDir disabled root hackerdude rbowen
```

  Or, better still, only enable for a few trusted users:

```
UserDir disabled
UserDir enabled rbowen sungo krietz
```

Chapter 10

Content Negotiation

http://httpd.apache.org/docs/mod/mod_negotiation.html

http://httpd.apache.org/docs/content-negotiation.html

Content negotiation is a server-side means of choosing for the user the document that best suits their preferences, as configured in their browser settings

This is accomplished by a combination of client-side settings and server-side configuration.

Content Negotiation does not translate documents. Try not to laugh when a student asks you this. It may seem absurd, but at least one student will ask how it manages to translate the documents into different languages, and how many languages it knows, or variations on that theme.

10.1 Client configuration

10.1.1 Accept* headers

Sent by the client to say what content types, and languages, they wish to receive. See content-negotiation.html (full URL above) for more complete discussion.

`Edit -> Preferences -> Navigator -> Languages` for language configuration options.

At this point, you should show the students the output of printenv, or another of the standard CGI programs that displays environment variables, and discuss the Accept* headers at length, making sure that they understand what each means. You should configure your browser to request several languages in order to make these headers more interesting.

10.1.2 Quality factors

Associated with each content type, these further specify what document types (language, content type, character set) are preferred over others.

10.2 Negotiation Methods

10.2.1 MultiViews

```
Options +MultiViews
```

Multiple variants of a particular document are placed in a directory, and `MultiViews` turned on for that directory. Following the algorithm described in the documentation, the file that most closely matches the preferences specified by the user is chosen and returned.

Example:

Client requests the resource `index`

In the directory, we have the following files:

```
index.html.en
index.html.fr
index.html.de
index.txt
index.pdf
```

If the client browser specifies that it prefers to receive documents in German, then this client will receive the document `index.html.de`, because it most closely matches the client's requirements.

Content negotiation via `MultiViews` is very slow, as it must get a directory listing in order to consider the files that match the name of the resource requested.

Important to note that the client can request the resource as `index` rather than `index.html`, in order to consider a wider range of possible variants of the file. If the resource `index.html` is requested instead, the files `index.txt` and `index.pdf` would not even be considered.

10.2.2 Type map files

Rather than leaving Apache to fend for itself, you can do some of the work for it by creating a type-map file, listing all variants of a particular document, and the information about these variants.

This file would be called either `example.var` or `example.html.var`, depending on whether you wanted to negotiate for the URL "example" or "example.html".

```
URI: example

URI: example.html.en
Content-type: text/html
Content-language: en

URI: example.html.fr
Content-type: text/html;charset=iso-8859-2
Content-language: fr
```

You can also specify content quality:

This file would be called `picture.var`

```
URI: picture

URI: picture.jpg
Content-type: image/jpeg; qs=0.8

URI: picture.gif
Content-type: image/gif; qs=0.5

URI: picture.txt
Content-type: text/plain; qs=0.01
```

The scenario here is that you have a particular picture available in a high-quality jpeg image, a low-quality gif image, and as ASCII art. Depending on the browser's preferences, they may get one version or another. Presumably, a text-only browser which rejects images entirely could still get the ASCII art version of the image, and still (sort of) appreciate the experience.

You'll need the following to enable this:

```
AddHandler type-map .var
```

10.3 Caching

Caching can be problematic because it may mean that a client might get a document that was right for someone else, but not for themselves.

For example, Pierre down the hall goes to CNN.com first thing in the morning, and gets the site in French, which is then cached. The rest of the day, everyone else keeps getting it in French, because they are getting it from the caching proxy server.

Note, however, that CacheNegotiatedDocs is off by default, and it's unlikely that anyone will ever turn it on.

```
CacheNegotiatedDocs Off
```

Chapter 11

Indexing with mod_autoindex

http://httpd.apache.org/docs/mod/mod_autoindex.html

If a directory is requested, such as `http://www.rcbowen.com/imho/`, then Apache can ...

1. Serve an index file
2. Display a file listing
3. Product an error message

11.1 Options

The `Indexes` option turns on the ability to display a directory listing in the event that there is no index file in the directory.

```
<Directory /usr/local/apache/icons>
Options +Indexes
</Directory>
```

11.2 DirectoryIndex

This directive specifies the file that is to e served when a directory is requested. Multiple files can be listed in the priority order in which they are to be considered.

```
DirectoryIndex index.html index.php index.cgi
```

11.3 IndexOptions

The arguments to `IndexOptions` are as follows

You should demonstrate each of these options so that the students can see what happens Note that for the ones that are Apache 2.0 specific, you will need to switch over to that configuration file, which will confuse the students that are following along. Encourage them to try each of these options as well.

- None

  IndexOptions None causes a directory listing to be generated as a simple bullet-list of items with links to the file itself.

- FancyIndexing

  This option is necessary for all following ones. That is, turning on FancyIndexing enables the use of all other index options.

- DescriptionWidth

- NameWidth

  Allows you to set the number of characters available for the description of the file or directory. Or the name of the file or directory. Respectively.

- FoldersFirst

  Display the folders first in the listing, as people are used to seeing in various file managers.

- HTMLTable (Apache 2.0)

  The default directory index listing is not HTML-complient, because it contains formatting and image tags inside pre tags. Some people get uptight about stuff like this. Displaying the listing as an HTML table gets around this, and produces HTML-complient listings. The purpose of the pre tags is to get the columns to line up, and pre-dates the availablilty of HTML tables. This feature is only available in Apache 2.0, because nobody has cared to back-port it to 1.3.

- IconHeight, IconWidth

  Adds width= and height= tags to the HTML img tag to facilitate rapid page rendering.

- IconsAreLinks

  The icons are not usually links to the file. This option makes them such. However, it also put the large blue border around each icon, which is irritating.

- IgnoreClient

- SuppressColumnSorting

  These two options should be considered together, as it is easy to get confused which is which, and why you need both of them.

  IgnoreClient ignores arguments passed in the URL for reordering the entries by column. However, the links are still at the top of each column to re-order the entries.

  SuppressColumnSorting removes the links at the top of each column, but still will honor the arguments in a URL to reorder, if the user types it in, or if it is linked to explicitly.

  So, it really only makes sense to use both of these options, or neither. I'm not sure why anyone would feel the need to do this, but perhaps a particular ordering is required for some pages.

- ScanHTMLTitles

  For HTML files, the file is opened and the value of the ¡title¿ tag is placed into the description field for the directory listing. Note that this is a HUGE performance hit.

- SuppressHTMLPreamble

  Useful when using HeaderName. The HTML preamble (the head and body tags, as well as the "Index of /foo" text) are ommitted, and you can replace them with the contents of the HeaderName file instead. Otherwise, any head or body tags that you put in the HeaderName file will probably be ignored by the client.

- SuppressDescription
- SuppressIcon
- SuppressLastModified
- SuppressSize

  Don't display the column in question

- SuppressRules
- TrackModified

  Send a `Last-Modified` header which reflects the last time a file in this directory was modified. Otherwise, the `Last-Modified` header will always be the time of the request, because the resulting document is being generated fresh each time.

- VersionSort

  1.1, 1.2, 1.10 rather than 1.1, 1.10, 1.2

Other directives are:

- AddIcon
- AddIconByType
- AddIconByEncoding
- DefaultIcon
- AddDescription

  Note that this directive is a substring match, not a literal file name, or even a file extension. This can cause confusion if you happen to have multiple files with similar names, and wish to give them different descriptions. Thus:

  ```
  AddDescription "The Foo page" foo.html
  AddDescription "Other foo stuff" foo
  ```

  Will not work as desired, whereas:

  ```
  AddDescription "The Foo page" foo.html
  AddDescription "Other foo stuff" foo
  ```

  will. Get it?

11.4 Additional directives

In addition to the `IndexOptions`, there are a few other directives that allow you to adjust how directory listings are displayed.

11.4.1 HeaderName

Display a file as a header in the directory listing. If this file is HTML, you can make the listing appear in the same look as the rest of your site. See `SuppressHTMLPreamble`.

```
HeaderName header.html
```

11.4.2 ReadmeName

Might perhaps be better named `FooterName`. This displays the contents of a file at the bottom of the directory listing.

```
ReadmeName footer.html
```

11.4.3 IndexIgnore

List files that you don't want to show up in the directory listing.

```
IndexIgnore .htaccess *.swp *.tmp
```

11.5 Searching and sorting

11.5.1 Apache 1.3

- Can sort by Name, Modified, Size, and Description
- `http://server/directory?S=D` - Sort descending by size
- `http://server/directory?M=A` - Sort ascending by modified date
- Page 130

11.5.2 Apache 2.0

- Much more full-featured sorting and searching
- Page 131
- IndexOrderDefault

```
IndexOrderDefault Ascending Size
```

11.6 Examples

Page 133, 134

11.7 Security Concerns

- Get to documents that are not linked
- Security by obscurity is not really security at all

Chapter 12

Handlers and Filters

http://httpd.apache.org/docs/handler.html

http://httpd.apache.org/docs-2.0/filter.html

http://httpd.apache.org/docs-2.0/mod/mod_actions.html

12.1 Handlers

Handlers are functions in Apache modules which produce dynamic content in response to a URL request. They are usually configured via a `<Location>` directive, or can be mapped to a particular file type.

12.1.1 Configuration directives

AddHandler

Maps a handler to a particular file extension. That is, associates a particular action or process to a given file type.

```
AddHandler cgi-script cgi pl py
```

The above directive tells Apache to consider any program with a `cgi`, `pl`, or `py` extension to be a CGI program, execute it, and pass the resulting output back to the client.

This handler is provided by the `mod_cgi` module.

SetHandler

Much like the `AddHandler` directive, but specifies that all files (or URLs) in the given scope (usually a `Location`, `Files`, or `Directory` section is to be handled by the specified handler.

```
<Location /server-status>
    SetHandler server-status
</Location>
```

The `server-status` handler is provided by the module `mod_status`.

RemoveHandler

Removes the action of a handler from a specified file extension.

```
RemoveHandler .html
```

Action and Script

Provides for the creation of custom handlers, by mapping a file type to a CGI URL.

There are two ways that this can be handled.

You can map an action to a particular type (MIME type) of file using just the `Action` directive:

```
Action image/gif /cgi-bin/watermark.cgi
```

Or, you can map a file extension to an `Action` in two steps:

```
AddHandler my-handler .gif
Action my-handler /cgi-bin/watermark.cgi
```

These two techniques are essentially equivalent. The latter creates a named handler, and then defines that handler to be the specified CGI program.

The `Script` directive is a little bit different, and seldom used. `Script` specifies that a particular script is to be used every time a particular HTTP method is used in a request.

```
Script POST /cgi-bin/post.cgi
```

12.1.2 Standard handlers

The following are the standard handlers - that is, those handlers that are defined by modules that come standard with Apache.

default-handler

Defined by the Apache core (rather than by an extension module) this is the handler that deals with requests for files. This is the default manner of dealing with a URL request if there is nothing else special about it.

This is the handler that was in use in the URL mapping section yesterday.

send-as-is

Defined by mod_asis, the `send-as-is` handler sends a file without prefacing it with any headers. It is assumed that the file itself will contain the headers as part of the content of the file.

```
Status: 301 Now where did I leave that URL
Location: http://xyz.abc.com/foo/bar.html
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Lame excuses'R'us</TITLE>
</HEAD>
<BODY>
<H1>Fred's exceptionally wonderful page has moved to
<A HREF="http://xyz.abc.com/foo/bar.html">Joe's</A> site.
</H1>
</BODY>
</HTML>
```

The `send-as-is` handler is enabled using the `AddHandler` directive:

```
AddHandler send-as-is .asis
```

It is customary to name these pages `something.asis`, however, in the case of the above example, when advertising a

page that has moved, you may want to use a <Files> section to map the send-as-is handler to just a particular file.

cgi-script

Provided by mod_cgi, the cgi-script handler executes a program and returns the output of it to the client.

```
AddHandler cgi-script .cgi .pl
```

Use of this handler is permitted by use of the Options +ExecCGI directive.

imap-file

Sometimes a little hard to explain because it is completely archaic. There used to be server-side image maps (circa 1996) before client-side image maps came into wide-spread use.

This handler is provided by mod_imap, and you should consult the documentation for this module if you are interested in more detail. You are unlikely to ever use this handler.

server-info

Provided by mod_info, this handler provides detailed information about the server configuration and what modules are loaded.

Configured with a <Location section:

```
<Location /server-info>
SetHandler server-info
</Location>
```

server-status

Provided by mod_status, this handler provides a snapshot of server activity, displaying what each child process is doing.

See also the ExtendedStatus directive.

```
<Location /server-status>
SetHandler server-status
</Location>
```

http://httpd.apache.org/server-status/ for a good example of this in action.

server-parsed

Provided by `mod_includes`, this handler parses HTML pages looking for SSI (Server-Side Includes) directives. If found, it processes these directives and replaces them with the result of the action described in the directive.

See section on SSI for more information.

Use of this handler is permitted by using the `Includes` argument to `Options`.

type-map

Provided by `mod_negotiation`, this handler indicates that a particular file is a type map file, describing the variants of a particular document

```
AddHandler type-map .var
```

12.1.3 Custom handlers

Custom handlers can be created using the `Action` and `Script` directives described above.

For example, using the following configuration:

```
Action text/html /cgi-bin/footer.pl
```

you can add a footer to the bottom of every HTML page, using a CGI program that looks like:

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";

my $file = $ENV{PATH_TRANSLATED};

open FILE, "<$file";
print while <FILE>;
close FILE;
print qq~

FOOTER GOES HERE
~;
```

12.2 Filters

Filters are a major enhancement that comes with Apache 2.0, and one of the things that has been discussed since the very early days of talking about what would be in 2.0.

Filters give you the ability to chain actions on input or output. The classic example of this is SSI and CGI. For years, a frequently asked question on the mailing list and news groups has been whether you could put SSI directives in the output of a CGI program, and then have them processed correctly before the content was served to the client. The answer to this question is no, because you have only one shot at producing dynamic content. You decide which module will handle a particular request, and that's it. You can't have it both ways.

Filter chains change the answer to no. You can specify that particular content is passed through one or more additional filters on its way out to the client. The `Includes` filter happens to be an available filter. You can specify that your CGI-generated content will pass through the `Includes` filter on the way out the wire to the client, and those directives will be correctly processed and the desired content filled in.

```
Options +ExecCGI
AddHandler cgi-script cgi
AddOutPutFilter INCLUDES cgi
```

Alternately, you can specify that content be processed by more than one filter. In the example below, files with a `.shtml` extension are processed by the `INCLUDES` filter, and then compressed with the `DEFLATE` filter:

```
AddOutputFilter INCLUDES;DEFLATE shtml
```

Chapter 13

Performance tuning

http://httpd.apache.org/docs/misc/perf-tuning.html

http://httpd.apache.org/docs/programs/logresolve.html

http://httpd.apache.org/docs/programs/ab.html

13.1 Optimization, benchmarking and profiling

Optimize the right thing

People have a tendency to spend an inordinate amount of time optimizing the wrong thing. Like, for example, optimizing something that takes 5% of the time, ignoring the thing that takes 80% of the time. Or whatever. Benchmarking something gives you at least a clue as to what is taking all the time. We're talking here about client benchmarking, and not really about profiling your actual CGI code, which is more about programming than about managing Apache, and so is, thankfully, out of scope for this course.

Note that no matter what you do, the network will always be the bottleneck

Of course, that's not really true, but it's a good myth to keep in mind. If most of your users are dialup users, or home users in general, this may be the case. If most of your users are business users, this may not be the case, but the network will still be a very big portion of any user's performance experience.

13.2 ab

ApacheBench. Benchmarking for web content.

```
/usr/local/apache/bin/ab -n 1000 http://localhost/index.html
```

Make sure that everyone runs this, ponders the results, compares it to other students' output, and so on. Make sure that they look at all the fields, and know what they mean. This may seem like a silly exercise at the time, but most of this stuff will come in useful throughout the rest of this section, and the rest of the course.

`-k` flag for KeepAlive.

Can run this against other servers as well as your own. Try not to do this without the permission of the server administrator.

Some sites will detect this as an attack, and block your address. That would be annoying.

13.3 Perl

```
use Benchmark;
use LWP::Simple;

timethese($count, {
    'Slow' => '$content = get("http://server/slow.cgi");',
    'Fast' => '$content = get("http://server/fast.cgi")',
    'EvenFaster' => '$content = get("http://server/mod_perl_handler")',
});
```

- Gives comparative times for the two documents
- Note that this includes network time
- And, of course, this can be used to time any pieces of code that you are interested in comparing
- It will be interesting to come back to this code after we have covered mod_perl

13.4 Optimizing hardware

- More RAM
- If you have to swap, all bets are off. When in doubt, buy more RAM.
- Fast disk access. RAID is good.
- Faster CPU
- Pretty much the obvious stuff. Apache does not require any custom hardware.

13.5 Tuning configuration settings

13.5.1 HostnameLookups

```
HostNameLookups Off
```

- DNS lookups take a long time
- Used to be on by default
- Now is off by default. Leave it that way
- /usr/local/apache/bin/logresolve < /usr/local/apache/logs/access_log > report
- Do log resolution and reporting somewhere other than on your production web server

13.5.2 Symbolic links

- Allow symlinks to improve performance
- -FollowSymlinks requires that every file path get checked for symlinks. Not just on the file itself but on every directory leading up to it.
- SymLinksIfOwnerMatch requires not only this, but that we check ownership of every file along the way.
- For best performance, always use FollowSymLinks, and never use SymlinksIfOwnerMatch

13.5.3 .htaccess files

- Very very bad
- Must check for the existence and contents of a particular file for every directory in the path to the target.
- AllowOverride None
- Put directives in the main server configuration file

13.5.4 Negotiation

- It is slow. Turn it off if you don't need it
- On an increasingly global web, more and more sites will need this.
- Try running `ab` against `index.html` vs `index.html.en`

13.5.5 Caching and proxying

Apache ships with a caching proxy server which you can configure to cache incoming or outgoing requests.

mod_proxy

```
ProxyRequests On
CacheRoot /var/httpd/cache
```

Squid

```
http://www.squid-cache.org
```

Benefits of a caching proxy

- Faster retrieval of remote content

  If you put your organization behind a caching proxy server, and have your users proxy all fo their content through it, then commonly-fetched content will be cached, so that they will access it across the LAN, rather than across the Internet.

- Faster serving of content, sometimes

  Using mod_proxy and mod_rewrite, you can have incoming requests farmed out to a list of servers, and have this content served through the proxy server.

  ```
  http://apache13/mod/mod_rewrite.html#RewriteMap
  ```

Disadvantages

- Don't always have fresh content
- Don't always get the negotiated document that you really wanted
- May get incorrect data from cached CGI program output.

Note: These last two cases should actually never happen. With `CacheNegotiatedDocs Off` and properly configured CGI programs, these resources should always specify that they don't want to be cached.

13.5.6 mod_mmap_static

Map static files into memory

Replaced in 2.0 by mod_file_cache

```
MMapFile /usr/local/apache/htdocs/index.html
```

13.6 Process Creation

- MinSpareServers
- MaxSpareServers
- StartServers

When additional servers are needed, they are started on the following schedule. One is launched in the first second, 2 in the second, 4 in the third, and so on exponentially until 32 are being launched each second. This rate is maintained until the MinSpareServers requirement is again satisfied.

13.6.1 MaxRequestsPerChild

- 0 means never kill the child
- On Solaris, there are Apache leaks. Set this to something non-zero
- On Windows, never set this to anything other than 0 * KeepAlive * KeepAliveTimeout
- Setting this too high, (or too low) causes a loss of the performance benefit of KeepAlive

13.7 KeepAlive

- KeepAlive On
- KeepAliveTimeout

13.8 CGI/Other dynamic content

- The greatest bottleneck I have had has been bad code.
- Poor algorithms can cause code to work well in low-stress testing environments, but perform very poorly when faced with large amounts of real data.
- Test with realistic data.
- If the user hits ”reload” before a document loads the first time, you have just doubled the load.
- Split dynamic content over several steps if this improves performance

Chapter 14

CGI programming

http://httpd.apache.org/docs/howto/cgi.html

http://httpd.apache.org/docs/mod/mod_cgi.html

Today you are going to write a CGI program and get it working. This CGI program will parse form contents and put a record in a database. (Note, if DBI is not available, we'll put this in a text file or similar.)

14.1 Introduction - The CGI

- Provides an interface for arbitrary programs to provide content for web pages
- CGI spec: `http://hoohoo.ncsa.uiuc.edu/cgi/interface.html`
- CGI 2: `http://cgi-spec.golux.com/`
- Advantages of CGI
  - Easy to write
  - Easy to maintain
  - Readily available examples for download
- Disadvantages
  - Slow startup $\rightarrow$ slow runtime
  - No maintenance of state
  - Most of the 'readily available examples' are very badly written
- Alternatives to CGI
  - mod_perl
  - PHP
  - FastCGI
  - ASP

  - JSP
  - etc, etc, etc
- Each addresses the above problems in different ways, but with many of the same general concepts:
  - Persistent database connections
  - In-memory interpreter
  - Direct interface to the web server API

14.2 Apache configuration

- `Options +ExecCGI`
- `AddHandler cgi-script`
  - Adds cgi execution to particular files
- `SetHandler cgi-script`
  - Adds cgi execution to all files in the range
- `ScriptAlias`
  - This is the preferred method
  - Keep track of your CGI programs
  - Don't have to expose the mechanisms (.cgi) – See Jakob Nielsen

14.3 How a CGI program works

- Input

  Input comes in from the browser in several different formats
  - Environment variables:

```
SERVER_SOFTWARE
SERVER_NAME
GATEWAY_INTERFACE

SERVER_PROTOCOL
SERVER_PORT
REQUEST_METHOD
PATH_INFO
PATH_TRANSLATED
SCRIPT_NAME
QUERY_STRING
REMOTE_ADDR
REMOTE_HOST
AUTH_TYPE
REMOTE_USER
REMOTE_IDENT
```

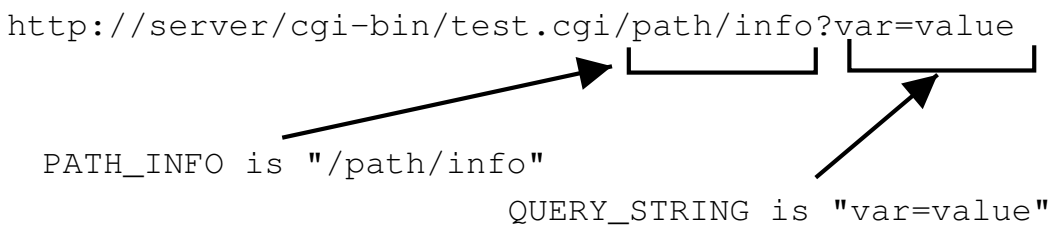
```
CONTENT_TYPE
CONTENT_LENGTH

HTTP_USER_AGENT
HTTP_ACCEPT
```

- `GET` requests

  Arguments following the end of the URL are available in the variables `QUERY_STRING` and `PATH_INFO`

```
http://server/cgi-bin/script.cgi/path/info?foo=bar&one=two
```



  `PATH_INFO` is `/path/info`

  `QUERY_STRING` is `foo=bar&one=two`

- Form input
- `POST` requests

  Form content has a similar format to `GET` information. It comes in over `STDIN`, and is formatted as `variable=value&variable=value`

- Decoding form data

  Most programming languages have some library available to decode form contents. We'll be using primarily Perl for the purpose of this tutorial, but other languages can be used.

- Output
- Content-type header
- Content

- Example programs

  Simple CGI programs in Perl

  Example 1

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello world";
```

  Example 2

```
#!/usr/bin/perl
use CGI;
my $cgi = new CGI;
$form = $cgi->Vars;

print "Content-type: text/html\n\n";
print "<h2>Form values ...</h2>";
foreach my $key (keys %$form) {
    print "$key => $form->{$key}<br>";
}
```

The above example needs an HTML form, something like:

```
<form action="/cgi-bin/example3.cgi" method="POST">
   One <input name="variableone" value="default"><br>
   Password <input type="password" name="pass"><br>
   <input type="hidden" name="x" value="y"><br>
   Checkbox <input type="checkbox" name="yesno"><br>
   Input area <textarea name="textarea" rows="5" cols="40"></textarea><br>
   <input type="submit" value="Submit Form">
</form>
```

Example in sh

```
#!/bin/sh
echo Content-type: text/html
echo
echo Hello, World
```

Example in C

```
#include <stdio.h>

int main()
    printf("Content-type: text/html\n\n");
    printf("Hello, world!\n");
    return 0;
```

14.4 Common problems

- Permissions
- Syntax errors
- Invalid headers
- Asking for help in a newsgroup

Chapter 15

SSI

http://httpd.apache.org/docs/howto/ssi.html

http://httpd.apache.org/docs/mod/mod_include.html

Directives written into the HTML pages, which are processed by the server as the page is being served.

- Add dynamic content to an existing HTML page
- Include external text files (headers, footers)

15.1 Configuration for SSI

- `Options +Includes`
- `Options +IncludesNOEXEC`

  Note that if you have `AllowOverride Options` enabled (or `AllowOverride All`, which includes `Options`) then people can put these directives in their .htaccess files, overriding your security precautions.
- Enabling by file extension

  ```
  AddType text/html .shtml
  AddHandler server-parsed .shtml
  ```

  This causes all files with a `.shtml` extension to be parsed for SSI directives.

  The disadvantage of this approach is that you have to

  - Expose the mechanism (ie, everyone knows how you are generating the effect)
  - You have the change the name of all the files, and break all the links to those files.

  You could also set all `.html` files to be parsed for SSI directives:

  ```
  AddHandler server-parsed .html
  ```

- Additional overhead on EVERY file
- Parsing files that have no directives in them
  Fortunately, there is an alternative.

15.2 XBitHack

The XBitHack directive tells Apache to parse files for SSI directives if they have the execute bit set.

```
XBitHack On
```

- on - parse files with u+x
- off - don't parse files with u+x
- full - g+x means send the last-modified date on the file itself, rather than the current time

Several advantages

- Don't have to change file names
- Don't expose the mechanism
- Guessable URLs
- Not supported on Windows - there's no x-bit

15.3 SSI directives

`Syntax: <!--#element attribute=value attribute=value ... -->`

Elements are ...

- `config`
- `echo`
- `exec`
- `fsize`
- `flastmod`
- `include`
- `printenv`

15.3.1 config

```
<!--#config errmsg="[It's broken]" -->
```

```
<!--#config sizefmt="bytes" --> (or abbrev)
```

15.3.2 timefmt

See page 201

```
<!--#config timefmt="%B %e, %Y" -->
```

15.3.3 echo

15.3.4 exec

- `cgi`
- `cmd`

15.3.5 fsize

- `file` (full path)
- `virtual` (URL path)

15.3.6 flastmod

15.3.7 include

15.3.8 printenv

15.4 Variables and flow control

See page 206-208

15.5 Security

- exec considered harmful
- Use 'include' rather than 'exec' for CGI programs
  - Removes security concerns with exec
  - Able to pass `QUERY_STRING` arguments

Part III

Day Three of Three

Chapter 16

Authentication, Authorization, Access Control

16.1 Definitions

- Authentication
- Authorization
- Access Control

16.2 Basic Authentication

- Provided by mod_auth
- 401 Authentication Required
- Browser supplies credentials if it has them
- Otherwise provides username/password dialog for user
- Credentials passed with every request
- Auth name/realm - used by the browser to cache login
- Password passed plaintext, with every request

16.3 Configuration

- Create a password file
  - htpasswd -c filename username
  - htpasswd filename username

- Set configuration to use this file
  - AuthType Basic
  - AuthName
  - AuthUserFile
  - Require user username
  - Require valid user
  - Use a different password than for your network login
- Optionally, create a group
  - group: user1 user2 user3
  - 8K limit
  - AuthGroupFile
  - Require group groupname

```
<Directory /usr/local/apache/htdocs/private>
  AuthType Basic
  AuthName "Top Sekrit"
  AuthUserFile /usr/local/apache/passwd/passwords
  Require user rbowen sungo
</Directory>
```

Or

```
<Directory /usr/local/apache/htdocs/private>
  AuthType Basic
  AuthName "Top Sekrit"
  AuthUserFile /usr/local/apache/passwd/passwords
  AuthGroupFile /usr/local/apache/passwords/groups
  Require group sekrit
</Directory>
```

`/usr/local/apache/passwords/groups` looks like:

```
    sekrit: rbowen sungo dpitts
```

16.4 FAQ

- How do I log out?
- How do I change what the password box looks like?
- How do I make my login persist across browser sessions?
- Why does it sometimes ask for my password twice?

16.5 Basic Auth Caveats

- Basic auth is not secure
- Username/password passed in the clear
- Content passed in the clear
- Cosmetic security only

16.6 Digest Auth

- Same as basic, except ...
- Username, password, MD5 hashed, and passed.
- Password not stored anywhere in the clear
- Content still passed in the clear
- Not supported by all browsers

Instead of htpasswd ...

```
htdigest -c /usr/local/apache/password/digest realm username
```

16.7 Configuration for Digest auth

```
AuthType Digest
AuthName "Private Area"
AuthDigestFile /usr/local/apache/passwords/digest
Require user drbacchus dorfl
```

Group file is identical to that used with Basic, if you want one. Use `AuthDigestGroupFile` with the same format.

16.8 Authentication against other things

- mod_auth_db
- mod_auth_mysql
- mod_auth_ldap
- mod_auth_nds (Netware Directory Services)
- mod_auth_smb (SMB - NT domain authentication)

16.8.1 mod_auth_db

- Creating a password file

```
dbmmanage passwords.dat adduser montressor
dbmmanage groups.dat add rbowen one,two,three
```

`dbmmanage --help` for full details, or `man dbmmanage`

This is still Basic authentication, with all the concerns pertaining thereto. It's just using a different file for its information.

```
AuthName "Members Only"
AuthType Basic
AuthDBUserFile /usr/local/apache/passwd/passwords.dat
AuthDBGroupFile /usr/local/apache/passwd/groups.dat
require group three
```

16.8.2 mod_auth_mysql

- User and password information in mysql
- Manage this information with whatever tools you're already using for database management.

16.9 Access Control

```
allow from address
deny from address
allow from 192.168
deny from dev.apacheadmin.com
deny from wanadoo.fr
```

The addresses specified can be a host name (partial, or complete) or an IP address (partial or complete).

To be even more specific, you need to use the Order directive:

```
Order deny,allow
Deny from all
Allow from apacheadmin.com
```

This can appear in a `<Directory` section, or in a `.htaccess` file.

These are applied as a series of filters. Everyone is excluded, then `apacheadmin.com` is let in. The other way around, it would be ineffectual.

Alternately,

```
Order allow,deny
Allow from all
Deny from wanadoo.fr
```

16.9.1 Satisfy

Use the satisfy directive when any one of a set of restrictions may be met. For example, if you want people inside your company to get into an area without being asked for a password, but people outside the company to be asked for a password, you could do the following:

```
Require group customers
Allow from internal.subnet.com
Satisfy any
```

See also `satisfy all` for another spin on this.

See also `mod_perl` access control handlers.

Chapter 17

Spiders

17.1 Introduction

- Indexing
- Searching
- Offline browsing
- Testing
- Link checking
- Performance testing (like ab)

17.2 Potential problems

- High server load
- Black holes
- DOS

17.3 Spiders in the logs

- altavista.com
- yahoo.com
- google.com
- etc
- Also, names like 'emailsiphon'

17.4 Excluding spiders from your site

There are a number of ways to exclude robots from your site.

17.4.1 robots.txt

Place a file called `robots.txt` in your `DocumentRoot` directory.

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /datafiles/
```

or

```
User-agent: Scooter
Disallow: /dont-index/
```

17.4.2 ROBOTS metatag

```
<META NAME="ROBOTS" CONTENT="INDEX,NOFOLLOW">
```

- INDEX
- NOINDEX
- FOLLOW
- NOFOLLOW

17.4.3 Yell at the operator

Look up the IP address that it is coming from, and email the admin at that location.

17.4.4 Block by address

```
Order allow,deny
Allow from all
Deny from unfriendly.spider.com
```

17.4.5 Blocking with Deny from Env

```
SetEnvIf User-Agent EmailSiphon Spammers
Order Allow,Deny
Allow from all
Deny from env=Spammers
```

17.5 Writing your own spider

- Don't
- Get one from somewhere else
- Writing a spider is easy
- Writing a *good* spider is hard
- See sourcecode in the book

Chapter 18

Security

http://httpd.apache.org/docs/misc/security-tips.html

18.1 Four main areas for security concerns

1. Apache itself
   - The last security hole found in Apache was in 1998
   - The Gartner Group recommends moving from IIS to Apache.
2. External attacks
3. Internal misconfiguration
4. Vendor security issues

18.2 Security guidelines

- Disable unused ports
- Remove unnecessary user accounts
- Don't use telnet
- Limit modules (Don't have modules installed that you are not using)
- Limit FrontPage
- Avoid SSI where not necessary
- Don't use the system password file for authentcation
- Don't put your password file in a document directory
- Develop on a staging server
- Keep up with OS and Apache security patches.

Chapter 19

Security in Dynamic Content

- SSI
- CGI
- mod_perl
- Wrappers
  - suexec
  - cgiwrap
- Logfiles and file permissions

Chapter 20

modules

20.1 Module list

These are the modules that come with Apache

20.1.1 Apache 1.3 modules:

(To get this moudle list, just type `./configure --help`

```
      [access=yes      actions=yes     alias=yes      ]
      [asis=yes        auth=yes        auth_anon=no   ]
      [auth_db=no      auth_dbm=no     auth_digest=no ]
      [autoindex=yes   cern_meta=no    cgi=yes        ]
      [digest=no       dir=yes         env=yes        ]
      [example=no      expires=no      headers=no     ]
      [imap=yes        include=yes     info=no        ]
      [log_agent=no    log_config=yes  log_referer=no ]
      [mime=yes        mime_magic=no   mmap_static=no ]
      [negotiation=yes proxy=no        rewrite=no     ]
      [setenvif=yes    so=no           speling=no     ]
      [status=yes      unique_id=no    userdir=yes    ]
      [usertrack=no    vhost_alias=no                 ]
```

20.1.2 Apache 2.0 modules:

```
[access         actions         alias      ]
[asis           auth            auth_anon  ]
[auth_dbm       auth_digest     autoindex  ]
[cache          cern_meta       cgi        ]
[cgid           charset_lite    dav        ]
[deflate        dir             env        ]
[example        expires         ext_filter ]
[file_cache     headers         imap       ]
[include        info            isapi      ]
[log_config     mime            mime_magic ]
[negotiation    proxy           rewrite    ]
[setenvif       so              speling    ]
[ssl            status          suexec     ]
[unique_id      userdir         usertrack  ]
[vhost_alias                               ]
```

20.1.3 What's new, and what's missing

The modules `mod_auth_db`, `mod_digest`, `mod_log_agent`, `mod_log_referer`, and `mod_mmap_static` go away in version 2.0.

New in version 2.0 are `mod_cache`, `mod_cgid`, `mod_charset_lite`, `mod_dav`, `mod_deflate`, `mod_ext_filter`, `mod_file_cache`, `mod_isapi`, and `mod_ssl`.

20.2 mod_access

```
Name            mod_access
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_access.html
```

mod_access provides access control based on client hostname, IP address, or other characteristics of the client request.

20.3 mod_actions

```
Name            mod_actions
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_actions.html
```

Provides for executing CGI scripts based on media type or request method.

20.4 mod_alias

```
Name            mod_alias
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_alias.html
```

Mapping different parts of the file system into the document tree, and URL redirection.

20.5 mod_asis

```
Name            mod_asis
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_asis.html
```

Sending files which contain their own HTTP headers.

```
Content-type: 32
Content-type: text/html
Content-language: en
Set-Cookie: name=value

And then the content goes here.
```

20.6 mod_auth

```
Name            mod_auth
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_auth.html
```

Basic HTTP authentication, using text files to contain user and group information.

20.7 mod_auth_anon

```
Name            mod_auth_anon
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_auth_anon.html
```

Anonymous user access to authenticated areas.

20.8 mod_auth_db

```
Name            mod_auth_db
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_auth_db.html
```

Basic HTTP authentication, using Berkeley DB files to contain user and group information.

This module is removed in Apache 2.0, and only `mod_auth_dbm` remains. There's very little chance that you'll ever have a need for both at the same time, or even have more than one dbm implementation install on the same machine.

20.9 mod_auth_dbm

```
Name            mod_auth_dbm
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_auth_dbm.html
```

Basic HTTP authentication, using DBM files to contain user and group information.

20.10 mod_auth_digest

```
Name            mod_auth_digest
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_auth_digest.html
```

MD5 digest authentication.

In Apache 2.0, no longer marked as experimental.

20.11 mod_autoindex

```
Name            mod_autoindex
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_autoindex.html
```

Automatic directory listings

20.12 mod_cern_meta

```
Name            mod_cern_meta
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_cern_meta.html
```

Support for HTTP header metafiles

20.13 mod_cgi

```
Name            mod_cgi
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_cgi.html
```

Support for execution of CGI programs

20.14 mod_digest

```
Name            mod_digest
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_digest.html
```

Provides MD5 authentication, but has been replaced by mod_auth_digest

20.15 mod_dir

```
Name            mod_dir
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_dir.html
```

Provides for mapping URLs with a trailing slash to an index file, typically called index.html

20.16 mod_env

```
Name            mod_env
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_env.html
```

Handles the passing of environment variables to CGI programs

20.17 mod_example

```
Name            mod_example
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_example.html
```

An example module, demonstrating the Apache API, and the technique of writing Apache modules

20.18 mod_expires

```
Name            mod_expires
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_expires.html
```

Gives the ability to apply Expires: headers to resources.

20.19 mod_headers

```
Name            mod_headers
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_headers.html
```

Add arbitrary HTTP headers to resources

20.20 mod_imap

```
Name            mod_imap
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_imap.html
```

Handles server-side image map files

20.21 mod_include

```
Name            mod_include
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_include.html
```

Server-parsed documents (Server-side includes)

20.22 mod_info

```
Name            mod_info
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_info.html
```

Provides the server-info handler, for providing information about server configuration.

20.23 mod_log_agent

```
Name            mod_log_agent
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_log_agent.html
```

Logging of user agent (browser). This module is superseded by the LogFormat directive in mod_log_config

20.24 mod_log_config

```
Name            mod_log_config
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_log_config.html
```

Allows you to build custom log files. See Chapter 24 for detailed treatment of this module and the functionality it provides.

20.25 mod_log_referer

```
Name            mod_log_referer
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_log_referer.html
```

Provides logging of document references. That is, logs the places that have links to your content. This module is superseded by the LogFornat directive in mod_log_config

20.26 mod_mime

```
Name            mod_mime
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_mime.html
```

Determining document types by file extensions. See Chapter 8.

20.27 mod_mime_magic

```
Name            mod_mime_magic
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_mime_magic.html
```

Determining document types using “magic numbers” - that is, by looking at the contents of the file, and, based on the frequency of occurance of certain patterns or characters, determining what the file type probably is.

20.28 mod_mmap_static

```
Name            mod_mmap_static
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_mmap_static.html
```

Mapping files into memory to improve performance of serving static document. This module is marked as experimental.

20.29 mod_negotiation

```
Name            mod_negotiation
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_negotiation.html
```

Content negotiation. See Chapter 10

20.30 mod_proxy

```
Name            mod_proxy
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_proxy.html
```

A caching proxy server.

20.31 mod_rewrite

```
Name            mod_rewrite
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_rewrite.html and
                http://httpd.apache.org/docs/misc/rewriteguide.html
```

Provides the ability to rewrite incoming URL requests in order to do all of the things that you wish mod_alias did.

20.32 mod_setenvif

```
Name            mod_setenvif
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_setenvif.html
```

Set environment variables based on client information. Can be used for things such as access control:

```
SetEnvIf User-Agent ^KnockKnock/2.0 let_me_in
<Directory /docroot>
Order Deny,Allow
    Deny from all
    Allow from env=let_me_in
</Directory>
```

20.33 mod_so

```
Name            mod_so
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_so.html
```

Dynamically load modules as shared objects at runtime.

20.34 mod_speling

```
Name            mod_speling
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_speling.html
```

Automatically correct minor typos in URLs, such as character transposing, wrong capitalization, or other small errors.

20.35 mod_status

```
Name            mod_status
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_status.html
```

Display server status in a convenient HTML report.

20.36 mod_unique_id

```
Name            mod_unique_id
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_unique_id.html
```

Generate unique identifiers for each incoming request for tracking purposes.

20.37 mod_usertrack

```
Name            mod_usertrack
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_usertrack.html
```

User tracking using cookies. Make sure you know what you are doing before enabling this. It sends a lot of cookies.

20.38 mod_vhost_alias

```
Name            mod_vhost_alias
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_vhost_alias.html
```

Dynamically configure a large number of virtual hosts without changing your server configuration file.

Chapter 21

mod_perl

21.1 Overview - What is mod_perl?

mod_perl embeds a Perl interpter into the Apache process, for two purposes:

- Make calls directly to the Apache API to write Apache modules in Perl
- Improve performance of Perl CGI programs upwards of 300%

21.2 Installation

Unpack mod_perl source, change into the mod_perl directory, and then ...

```
perl Makefile.PL APACHE_PREFIX=/usr/local/apache
APACHE_SRC=../apache-1.3.20/src DO_HTTPD=1 USE_APACI=1
EVERYTHING=1
APACI_ARGS='--enable-module=rewrite,--enable-module=speling'

make && make install
```

21.3 mod_perl installation caveats

- Don't install as dso
- php

You can install it in conjunction with PHP if you are really careful, but there are frequent problems with this interaction. In particular, they seem to use conflicting versions of the mysql libraries, and this can cause conflicts if/when they both attempt to connect to a mysql database.

21.4 Configuration

21.4.1 PerlRequire

Perl commands that you want to run at startup. Preload modules into shared memory. Set global variables.

```
PerlRequire /usr/local/apache/conf/preload.pl
```

And `/usr/local/apache/conf/preload.pl` would then contain:

```
use Apache::DBI;
use DBI;
use CGI qw(:Standard);
use MyCompany::Utils;
use lib '/path/to/my/modules';
1;
```

21.5 Connecting to your database

```
Apache::DBI->connect_on_init( $database, $username, $password );
```

21.6 CGI under mod_perl

A major use of mod_perl is as a CGI speed enhancer. This can provide 10 to 20 times speed improvement, in my experience.

This is done one of two ways.

21.6.1 Apache::PerlRun

If you have cgi code that works and you don't want to spend much time ensuring that it is safe to run under mod_perl, you can just use Apache::PerlRun to run these programs and gain some of the benefits of mod_perl.

```
Alias /cgi-perl/ /usr/local/apache/cgi-bin/
<Location /cgi-perl>
    SetHandler perl-script
    PerlHandler Apache::PerlRun
    Options ExecCGI
    PerlSendHeader on
</Location>
```

Maps the URL `/cgi-perl` to your cgi-bin directory, and arranges for mod_perl to execute these cgi programs for you, rather than `mod_cgi`. You can now access your cgi programs with a new URL, and get an immediate speed improvement.

Rather than using the URL

```
http://servername/cgi-bin/test.cgi
```

you can now use the URL

```
http://servername/cgi-perl/test.cgi
```

For the purpose of this exercise, we will use the following CGI program, and call it `test.cgi`. Please type in this program and place it in `/usr/local/apache/cgi-bin`:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello";
```

You now should run the following command:

```
/usr/local/apache/bin/ab -n 1000 -c 5 http://localhost/cgi-bin/test.cgi
```

After noting the numbers that you get, then run the following:

```
/usr/local/apache/bin/ab -n 1000 -c 5 http://localhost/cgi-perl/test.cgi
```

Wow.

21.6.2 Apache::Registry

If you are certain that your program is well-written, uses `strict` and `warnings`, and does not abuse global varibles, try it under `Apache::Registry`

Add the following configuration:

```
Alias /perl/ /usr/local/apache/cgi-bin/
<Location /perl>
    SetHandler perl-script
    PerlHandler Apache::Registry
    Options +ExecCGI
    PerlSendHeader on
</Location>
```

Now, run:

```
/usr/local/apache/bin/ab -n 1000 -c 5 http://localhost/perl/test.cgi
```

Be impressed.

21.7 Apache handlers with mod_perl

If you really want to take advantage of the power of `mod_perl`, you should write Apache handlers using mod_perl.

21.7.1 Installing a mod_perl handler from CPAN

There are a plethora of existing mod_perl handlers available for download from CPAN (the Comprehensive Perl Archive Network - http://www.cpan.org/). We'll install one, and talk about another one.

Install `Apache::Perldoc` using:

```
# perl -MCPAN -e shell

cpan> install Apache::Perldoc
```

Configure the module using:

```
<Location /perldoc>
    SetHandler perl-script
    PerlHandler Apache::Perldoc
</Location>
```

And then use the module by going to the URL

```
http://localhost/perldoc/Apache::Perldoc
```

Generates full documentation for any Perl module that you have installed.

`Apache::Album` allows you to put image files on your server, and automatically generate image galleries with thumbnail images:

```
http://buglet.rcbowen.com/photos/
```

21.8 Writing a mod_perl handler

A mod_perl handler is a Perl module with a single method called `handler`. This method should take a single argument - the `Apache::Request` object, traditionally called `$r`, and should emit content to get displayed in the browser, using the appropriate methods from the Apache API

21.8.1 Example mod_perl handlers

```
package Apache::HandlerTest;

sub handler {
    my $r = shift; # Apache session object
    $r->content_type('text/html');
    $r->send_http_header;
    $r->print( "Hello, world." );
}
```

21.8.2 Installing the example mod_perl handler

Because Perl looks certain places for Perl modules, this module needs to be placed in the Perl library directory. There are a variety of ways to do this, and for the purpose of this course, we will just copy the file into the Perl library directory manually.

The above file is to be called `HandlerTest.pm`, and is to be placed in an `Apache` subdirectory of the Perl lib directory.

For example, if the Perl lib directory is
`/usr/lib/perl5/site_perl/5.6.0`
then the file should be placed at
`/usr/lib/perl5/site_perl/5.6.0/Apache/HandlerTest.pm`

If you don't know much about Perl, ask the instructor for assitance at this point.

21.8.3 Configuring the mod_perl handler

The handler is configured by adding a `<Location>` section in your configuration:

```
<Location /handlertest>
    SetHandler perl-script
    PerlHandler Apache::HandlerTest
</Location>
```

21.9 Common problems

21.9.1 Don't exit

Calling the Perl command `exit()` causes the Perl interpreter to exit, rendering that Apache child useless. Don't do that.

21.9.2 Restart the server

When you restart, you actually have to stop and start the server, as your code is often cached in the parent process, and so restarting the child processes does not cut it.

See also:

- PerlFreshRestart
- Apache::StatINC

21.9.3 Global values

Global values in `mod_perl` are really global. Meaning that not only can variables be seen out of scope, but they can also be seen in other child processes (maybe) and in other client accesses. This can really ruin your whole day.

21.10 More information

http://perl.apache.org/

`mod_perl` Developer's Cookbook (Geoff Young)

Chapter 22

SSL

http://openssl.org/

http://modssl.org/

- public/private key cryptography
- SSL Certificates
- Signed certificates
  - Thawte
  - Verisign
  - Sign it yourself

22.1 Intro

- Runs on port 443
- Accessed via https://servername

22.2 Installing SSL

- OpenSSL
- mod_ssl
- Apache 2.0 - ./configure –with-ssl

22.3 Certificates

- Generate a key pair

```
openssl genrsa -rand file1:file2:file3
-out www.domain.com.key 1024
```

Don't enter a pass phrase, or you will have to enter that passphrase every time you restart Apache

- Generate certificate signing request

```
openssl req -new -key www.domain.com.key -out www.domain.com.csr
```

You will be asked for a series of information, to which you need to provide answers. These answers will appear on the certificate itself, and verify its authenticity.

Note that when it asks for your 'Common Name', this is actually the name of the web site - the fully qualified domain name which will be used to serve the site. So this should be, for example, 'www.rcbowen.com' or similar. Not your name. Not your company name.

At this point, you can either sign this yourself, or you can submit the certificate request to one of the CA (Certificate Authorities) for signing.

`http://digitalid.verisign.com/server/apacheNotice.htm`

`http://www.thawte.com/certs/server/request.html`

- Sign it yourself

```
openssl x509 -req -days 365 -in www.domain.com.csr -signkey
www.domain.com.key -out www.domain.com.cert
```

- Install the key - place it in the certs directory (should be either under the openssl directory, or somewhere under the Apache directory.

22.4 Configuration

```
<VirtualHost _default_:443>
ServerName www.domain.com
SSLEngine on
SSLCertificateFile /path/to/www.comain.com.cert
SSLCertificateKeyFile /path/to/www.domain.com.key
</VirtualHost>
```

See also the default SSL configuration that comes with openssl when you install it. It has a lot more stuff than this in it.

Note that you can also use *:443 instead, but, if you do, you need to make sure that all the other vhosts that you are running use the same syntax.

- mod_speling
- mod_rewrite
- Migrating to Apache 2.0